WRDC-TR-90-8007
Volume VIII
Part 5

INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume VIII - User Interface Subsystem
Part 5 - Forms Processor Development Specification

S. Barker

Control Data Corporation
Integration Technology Services
2970 Presidential Drive
Fairborn, OH 45324-6209

September 1990

Final Report for Period 1 April 1987 - 31 December 1990

MANUFACTURING TECHNOLOGY DIRECTORATE
WRIGHT RESEARCH AND DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6533

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, regardless whether or not the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data. It should not, therefore, be construed or implied by any person, persons, or organization that the Government is licensing or conveying any rights or permission to manufacture, use, or market any patented invention that may in any way be related thereto.

This technical report has been reviewed and is approved for publication.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations

DAVID L. JUDSON, Project Manager
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

FOR THE COMMANDER:

BRUCE A. RASMUSSEN, Chief
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

If your address has changed, if you wish to be removed form our mailing list, or if the addressee is no longer employed by your organization please notify WRDC/MTI, Wright-Patterson Air Force Base, OH 45433-6533 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution is Unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) DS 620344200	5. MONITORING ORGANIZATION REPORT NUMBER(S) WRDC-TR-90-8007 Vol. VIII, Part 5
6a. NAME OF PERFORMING ORGANIZATION Control Data Corporation; Integration Technology Services 6b. OFFICE SYMBOL *(if applicable)*	7a. NAME OF MONITORING ORGANIZATION WRDC/MTI
6c. ADDRESS *(City,State, and ZIP Code)* 2970 Presidential Drive Fairborn, OH 45324-6209	7b. ADDRESS *(City, State, and ZIP Code)* WPAFB, OH 45433-6533
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Wright Research and Development Center, Air Force Systems Command, USAF 8b. OFFICE SYMBOL *(if applicable)* WRDC/MTI	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUM. F33600-87-C-0464
8c. ADDRESS *(City, State, and ZIP Code)* Wright-Patterson AFB, Ohio 45433-6533	10. SOURCE OF FUNDING NOS.
11. TITLE See block 19	PROGRAM ELEMENT NO. 78011F / PROJECT NO. 595600 / TASK NO. F95600 / WORK UNIT NO. 20950607
12. PERSONAL AUTHOR(S) Structural Dynamics Research Corporation: Barker, S., et al.	
13a. TYPE OF REPORT Final Report 13b. TIME COVERED 4/1/87-12/31/90	14. DATE OF REPORT *(Yr.,Mo.,Day)* 1990 September 30 15. PAGE COUNT 70
16. SUPPLEMENTARY NOTATION WRDC/MTI Project Priority 6203	
17. COSATI CODES FIELD: 1308 / GROUP: 0905 / SUB GR.	18. SUBJECT TERMS *(Continue on reverse if necessary and identify block no.)*

19. ABSTRACT *(Continue on reverse if necessary and identify block number)*

This specification establishes the conceptual design of the system identified as the Form Processor.

BLOCK 11:

INTEGRATED INFORMATION SUPPORT SYSTEM
Vol VIII - User Interface Subsystem

Part 5 - Forms Processor Development Specification

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED x SAME AS RPT. DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL David L. Judson	22b. TELEPHONE NO. *(Include Area Code)* (513) 255-7371 22c. OFFICE SYMBOL WRDC/MTI

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE

FOREWORD

This technical report covers work performed under Air Force Contract F33600-87-C-0464, DAPro Project. This contract is sponsored by the Manufacturing Technology Directorate, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Bruce A. Rasmussen, Branch Chief, Integration Technology Division, Manufacturing Technology Directorate, through Mr. David L. Judson, Project Manager. The Prime Contractor was Integration Technology Services, Software Programs Division, of the Control Data Corporation, Dayton, Ohio, under the direction of Mr. W. A. Osborne. The DAPro Project Manager for Control Data Corporation was Mr. Jimmy P. Maxwell.

The DAPro project was created to continue the development, test, and demonstration of the Integrated Information Support System (IISS). The IISS technology work comprises enhancements to IISS software and the establishment and operation of IISS test bed hardware and communications for developers and users.

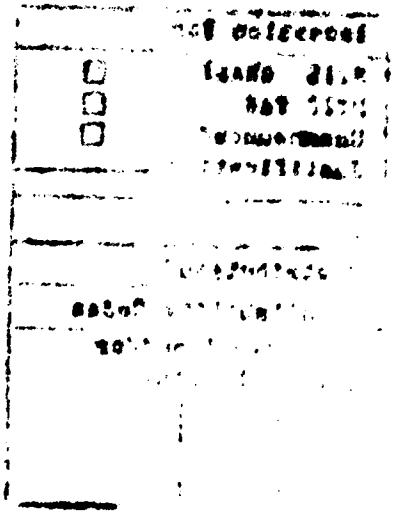
The following list names the Control Data Corporation subcontractors and their contributing activities:

SUBCONTRACTOR	ROLE
Control Data Corporation	Responsible for the overall Common Data Model design development and implementation, IISS integration and test, and technology transfer of IISS.
D. Appleton Company	Responsible for providing software information services for the Common Data Model and IDEF1X integration methodology.
ONTEK	Responsible for defining and testing a representative integrated system base in Artificial Intelligence techniques to establish fitness for use.
Simpact Corporation	Responsible for Communication development.
Structural Dynamics Research Corporation	Responsible for User Interfaces, Virtual Terminal Interface, and Network Transaction Manager design, development, implementation, and support.
Arizona State University	Responsible for test bed operations and support.

This report includes the following topics:

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS



SECTION 1

SCOPE

1.1 Identification

This specification establishes the conceptual design of the system identified as the Form Processor. This system will contain the Graphics Support System (GSS). This system is a subsystem of the User Interface System (UIS) which provides both two and three dimensional graphics support for the IISS testbed.

NOTE: The conceptual design presented in this specification is intended to encompass all of the desired characteristics of the ultimate system. It is not intended to imply that all of these capabilities will be implemented as a part of the current task or of any future task. An attempt has been made to identify those using phrases such as "eventually" or "in the future" when referring to them within this specification. Additional determinations of applicability and scope will be made as design proceeds and presented in subsequent life cycle documents.

1.2 Functional Summary

One of the objectives of the IISS testbed is to allow applications to be run from a wide variety of terminals using formatted screens for input and output of application data. Instead of the application programs having to contain terminal dependent code to send and receive formatted screens to and from various types of terminals and to perform terminal control functions, the program may use the set of callable execution time routines of the FP.

The Graphics Support System (GSS) is a computing subsystem of the User Interface System of IISS which provides two and three dimensional graphics along with high level icon and business graph support. The high level support is accomplished through the use of compilers for static definitions and callable routines for dynamic definitions. The GSS uses the Programmer's Hierarchical Interactive Graphics System (PHIGS) as its basics. The IISS user may access the PHIGS routines directly in order to display graphic images.

The major functions provided by the FP are:

- Form Processing
- User Profile Maintenance
- Virtual Terminal Pass-through
- NTM Message Processing
- Function Key Processing
- Scripting
- Two and Three Dimensional Graphics Processing

The details of these functions are explained in Section 3.2 Detailed Functional Requirements.

SECTION 2

DOCUMENTS

2.1 Reference Documents

[1] General Electric Co., ICAM Integrated Support System (IISS) Test Bed System Design Specification (Draft), 7 Feb 83, SDS620140000.

[2] Systran, ICAM Documentation Standards, 15 September 1983, IDS150120000C.

[3] Structural Dynamics Research Corporation, Form Processor User Manual, UM 620244200A, 16 February 1987.

[4] Structural Dynamics Research Corporation, Report Writer Development Specification, DS 620244501A, 16 February 1987.

[5] Structural Dynamics Research Corporation, Rapid Application Generator Development Specification, DS 620244502A, 16 February 1987.

[6] Structural Dynamics Research Corporation, Text Editor Development Specification, DS 620244600A, 16 February 1987.

[7] Structural Dynamics Research Corporation, Form Processor Development Specification, DS 620244200 16 February 1987.

[8] Structural Dynamics Research Corporation, Application Interface Development Specification, DS 620244700A, 16 February 1987.

[9] Structural Dynamics Research Corporation, Forms Language Compiler Development Specification, DS 620244401B, 8 December 1987.

[10] Structural Dynamics Research Corporation, Forms Driven Form Editor Development Specification, DS 620244402A, 16 February 1987.

[11] Structural Dynamics Research Corporation, User Interface Services Development Specification, DS 620244100A, 16 February 1987.

[12] Structural Dynamics Research Corporation, Virtual Terminal Development Specification, DS 620244300A, 16 February 1987.

[13] General Electric Corporation, IISS System Design Specification, SDS 620140000, 7 February 1983.

[14] Structural Dynamics Research Corporation, IISS Form Processor Application Interface, DS 620244700A, 16 February 1987.

[15] Structural Dynamics Research Corporation, Form Editor User's Manual, UM 620244400A, 16 February 1987.

[16] Structural Dynamics Research Corporation, Rapid Application Generator User's Manual, UM 620244502A, 16 February 1987.

[17] American National Standards Institute, Programmer's Hierarchical Interactive Graphics System, dpANS X3.144 (Draft Proposed Standard X3H3/87-100).

[18] American National Standards Institute, Graphical Kernel System Functional Description, ANSI X3.124-1985.

[19] International Organization for Standardization, Graphical Kernel System (GKS) functional description, ISO 7942-1985.

[20] American National Standards Institute, Computer Graphics Metafile for the Storage and Transfer of Picture Description Information, ANSI X3.122-1986.

[21] Structural Dynamics Research Corporation, C Coding Guidelines, IISS Programmer's Guide

[22] American National Standards Institute, Additional Controls for use with American National Standard Code for Information Interchange, ANSI X3.64-1975.

2.2 Terms and Abbreviations

American Standard Code for Information Interchange: (ASCII), the character set defined by ANSI X3.4 and used by most computer vendors.

Application Generator (AG): A subset of the IISS User Interface that consists of software modules that generate IISS application code and associated form definitions based on a language input. The part of the AG that generates report programs is called the Report Writer. The part of the AG that generates interactive applications is called the Rapid Application Generator.

Application Interface: (AI), subset of the IISS User Interface that consists of the callable routines that are linked with applications that use the Form Processor or Virtual Terminal. The AI enables applications to be hosted on computers other than the host of the User Interface.

Application Process: (AP), a cohesive unit of software that can be initiated as a unit to perform some function or functions.

Archive: A file containing the definitions of one or more structures. Structures can be saved into or retrieved from an archive by means of subroutine calls.

Attribute: field characteristic such as blinking, highlighted, black, etc. and various other combinations. Background attributes are defined for forms or windows only. Foreground attributes are defined for items. Attributes may be permanent, i.e., they remain the same unless changed by the application program, or they may be temporary, i.e., they remain in effect until the window is redisplayed.

Cell Array: A geometric primitive which consists of a number of adjoining colored or shaded parallelograms.

Closed Figure: A figure is closed if the path traced by a moving point returns to its starting position. The starting position may be arbitrarily assigned. "Fillarea" is synonymous with "closed figure".

Common Data Model: (CDM), IISS subsystem that describes common data application process formats, form definitions, etc. of the IISS and includes conceptual schema, external schemas, internal schemas, and schema transformation operators.

Complex Figure: A figure is complex if the path traced by a moving point crosses itself. An arbitrary point may be determined to be contained within the traced boundary if a line drawn to infinity crosses the boundary an odd number of times. If the number of crossings is zero or even, the point is outside the traced boundary.

Computer Graphics Metafile (CGM): A file with a standardized format which is used to store or transmit graphic images.

Computer Program Configuration Item: (CPCI), an aggregation of computer programs or any of their discrete portions, which satisfies an end-use function.

Conceptual Schema: (CS), the standard definition used for all data in the CDM. It is based on IDEF1 information modeling.

Current Cursor Position: the position of the cursor before an edit command or function is issued in the text editor.

Cursor Position: the position of the cursor after any command is issued.

Dependent Data: Data correlated to a dependent variable.

Dependent Variable: A mathematical variable whose value is determined by that of one or more other variables in a function.

Device Drivers: (DD), software modules written to handle I/O for a specific kind of terminal. The modules map terminal specific commands and data to a neutral format. Device Drivers are part of the UI Virtual Terminal.

Display List: An internal Form Processor list that contains only those forms that have been added to the screen and are currently displayed on the screen, along with information on where those forms are used.

Display Size: the number of lines used in the edit area.

Element: A graphics line or other primitive composed of graphics lines, such as an arc.

Extended Binary Coded Decimal Interchange Code: (EBCDIC), the character set used by a few computer vendors (notably IBM) instead of ASCII.

External Schema: (ES), an application's view of the CDM's conceptual schema.

Field: In reference to the Forms Processor, "field" refers to any object on the open or display list. These objects can be forms, items, window, etc. In reference to graphs, "field" refers to a collection of one or more graph figures. A graph field can be an axis, curve, pie chart, grid, etc.

Field Pointer: indicates the ITEM which contains the current cursor position.

Figure: A collection of elements. A figure may be closed or open.

Fill Area: A geometric primitive consisting of a planar area which is to be filled in with a particular color or pattern.

Form: structured view which may be imposed on windows or other forms. A form is composed of fields. These fields may be defined as forms, items, and windows, prompts, non-graphics lines, and structures.

Form Definition: (FD), form definition language after compilation. It is read at runtime by the Form Processor.

Form Definition Language: (FDL), the language in which electronic forms are defined.

Forms Driven Form Editor: (FDFE), subset of the Form Editor which consists of a forms-driven application used to create Form Definition files interactively.

Form Editor: (FE), subset of the IISS User Interface that is used to create definitions of forms. The FE consists of the Forms Driven Form Editor (FDFE) and the Forms Language Compiler (FLAN).

Form Hierarchy: a graphic representation of the way in which forms, items and windows are related to their parent form.

Form Language Compiler: (FLAN), subset of the Form Editor that consists of a batch process that accepts a series of form definition language (FDL) statements and produces form definition files as output.

Form Processor: (FP), subset of the IISS User Interface that consists of a set of callable execution time routines available to an application program for form processing.

Form Processor Text Editor: (FPTE), subset of the Form Processor that consists of software modules that provide text editing capabilities to all users of applications that use the Form Processor.

Generalized Drawing Primitive (GDP): A geometric primitive whose exact definition and representation is not specified. This allows an implementation to support additional geometric primitives such as arcs or conic sections in a standard conforming manner.

Graph: A picture correlated with data that alters as the data changes; by necessity, this is a dynamic (not predefined) picture. A graph may be imposed on windows or forms.

Graph Definition Language (GDL): An extension of the Forms Definition Language (FDL) which is used to define business graphs such as pie charts, X-Y plots, and bar charts.

Graph Figure: A collection of graphics primitives. The primitives can be circles, lines, arcs, etc.

Graphical Kernel System (GKS): A 2-dimensional graphics standard which is defined independently of any programming language.

Icon: A collection of figures and points that is predefined. An icon may be imposed on windows and forms. "Icon" is synonymous with "picture".

IISS Function Screen: the first screen that is displayed after logon. It allows the user to specify the function he wants to access and the device type and device name on which he is working.

Independent Data: Data that is correlated to an independent variable.

Independent Variable: A mathematical variable whose value is specified first and determines the value of one or more other values in an expression or function. For example, in a business graph of sales versus month, month is the independent variable and sales is the dependent variable, because sales varies by month.

Integrated Information Support System: (IISS), a test computing environment used to investigate, demonstrate and test the concepts of information management and information integration in the context of Aerospace Manufacturing. The IISS

addresses the problems of integration of data resident on heterogeneous data bases supported by heterogeneous computers interconnected via a Local Area Network. (LAN).

Item: non-decomposable area of a form in which hard-coded descriptive text may be placed and the only defined areas where user data may be input/output.

Local Area Network (LAN): A privately owned network that offers reliable, high-speed communications channels optimized for connecting information processing equipment in a limited geographic area.

Logical Device: a conceptual device that identifies a top level window of an application. It is used to distinguish between multiple applications running simultaneously on a physical device. NOTE that a single application can have more than one logical device. To the end user, this also appears as multiple applications running simultaneously.

Message: descriptive text which may be returned in the standard message line on the terminal screen. They are used to warn of errors or provide other user information.

Message Line: a line on the terminal screen that is used to display messages.

Network Transaction Manager: (NTM), IISS subsystem that performs the coordination, communication and housekeeping functions required to integrate the Application Processes and System Services resident on the various hosts into a cohesive system.

Open Figure: A figure is open if the path traced by a moving point does not return to its starting position. The starting position may be arbitrarily assigned. "Polyline" is synonymous with "open figure".

Open List: An internal Form Processor list that contains all forms that the application has opened for use along with information on where the form is used.

Operating System: (OS), software supplied with a computer which allows it to supervise its own operations and manage access to hardware facilities such as memory and peripherals.

Page: An instance of form in a window that is created whenever a form is added to a window.

Paging and Scrolling: a method which allows a form to contain more data than can be displayed with provisions for viewing any portion of the data buffer.

Physical Device: a hardware terminal.

Picture: A collection of figures and points that is predefined. A picture may be imposed on a window or a form. "Picture" is synonymous with "icon".

Polyline: A geometric primitive consisting of one or more connected line segments.

Polymarker: A geometric primitive consisting of one or more marker symbols (such as a cross or a dot).

Presentation Schema: (PS), may be equivalent to a form. It is the view presented to the user of the application.

Previous Cursor Position: the position of the cursor when the previous edit command was issued.

Programmer's Hierarchical Interactive Graphics System (PHIGS): A two and three dimensional graphics draft standard which is defined independently of any programming language.

Qualified Name: the name of a form, item or window preceded by the hierarchy path so that it is uniquely identified.

Report Definition Language: an extension of the Forms Definition Language that includes retrieval and calculation of database information and is used to define reports.

Structure: A collection of graphic primitives much as a form is a collection of textual primitives.

Subform: a form that is used within another form.

Text: A geometric primitive consisting of a number of characters with a particular orientation arranged along a particular path and aligned in a particular manner with some point. In PHIGS, text may be specified as a part of the image that participates fully in all transformations or as annotation which remains in the plane of the screen at all times.

Text Editor (TE): A subset of the IISS User Interface that consists of a file editor that is based on the test editing functions built into the Form Processor (FP).

User Data: Data which is either input by the user or output by the application programs to items.

User Interface: (UI), A subsystem of IISS that controls the user's terminal and interfaces with the rest of the system. The UI consists of two major subsystems: the User Interface Development System (UIDS) and the User Interface Management System (UIMS).

User Interface Development System: (UIDS), collection of IISS User Interface subsystems that are used by applications programmers as they develop IISS applications. The UIDS includes the Form Editor (FE) and the Application Generator (AG).

User Interface Management System: (UIMS), the runtime UI. It consists of the Form Processor (FP), Virtual Terminal (VT), Application Interface (AI), the User Interface Services (UIS) and the Text Editor (TE).

User Interface Monitor: (UIM), part of the Form Processor that handles messaging between the NTM and the UI. It also provides authorization checks and initiates applications.

User Interface Services (UIS): A subset of the IISS User Interface that consists of a package of routines that aid users in controlling their environment. It included message management, change password, and application definition services.

User Interface/Virtual Terminal Interface: (UI/VTI), another name for the User Interface.

Virtual Terminal: (VT), subset of the IISS User Interface that performs the interfacing between different terminals and the UI. This is done by defining a specific set of terminal features and protocols which must be supported by the UI software which constitutes the virtual terminal definition. Specific terminals are then mapped against the virtual terminal software by specific software modules written for each type of real terminal supported.

Virtual Terminal Interface (VTI): The protocol used to communicate with a device driver.

Window: dynamic area of a terminal screen on which predefined forms may be placed at run time.

Window Manager: a facility which allows the following to be manipulated: size and location of windows, the device on which an application is running, the position of a form within a window. It is part of the Form Processor(FP).

Workstation: an abstract graphical workstation which provides the logical interface to the applications program. It is analogous to a form in a window.

SECTION 3

REQUIREMENTS

This section includes functional and performance requirements for the FP. In addition, the FP interfaces to other IISS Computer Program Configuration Item's (CPCI's) are defined.

3.1 Computer Program Definition

3.1.1 System Capacities

The Form Processor supports simultaneous users each of which may be running multiple applications and each of these applications may require many forms. The maximum number of users, applications, and forms is limited only by external constraints such as the amount of available memory and the number of available NTM logical channels.

The following minimum PHIGS capabilities will eventually be supported:

Foreground Colors	1
Linetypes	4
Linewidths	1
Predefined polyline bundles	5
Settable polyline bundles	20
Marker types	5
Marker sizes	1
Predefined polymarker bundles	5
Settable polymarker bundles	20
Character heights	1
Character expansion factors	1
Character sets	1
String precision fonts	1
Character precision fonts	1
Stroke precision fonts	2
Predefined text bundles	6
Settable text bundles	20
Predefined interior bundles	5
Settable interior bundles	20
Predefined edge bundles	5
Settable edge bundles	20
Edgetypes	1
Edgewidths	1
Predefined patterns (see note 1)	1
Settable patterns (see note 1)	10
Hatch styles (see note 2)	3
Predefined view table entries (see note 3)	6
Settable view table entries	5
Structure priorities	2
Input classes	6
Prompt and echo types per device	1
Length of input queue (see note 4)	20
Maximum string buffer size (characters)	72
Maximum stroke buffer size (points)	64

Workstations of category OUTPUT OR OUTIN	1
Workstations of category INPUT or OUTIN	1
MO workstations	0
MI workstations	0
Archive files	1
HLHSR identifiers	1
Modeling clipping operators	3
Modeling clipping half-spaces	6
Annotation styles	2

NOTES:

1) relevant only for workstations supporting PATTERN interior style.
2) relevant only for workstations supporting HATCH interior style.
3) view table entry 0 is always defined, cannot be changed, and is set to the default values.
4) since available resources are finite and entries have variable size, it may not always be possible to achieve the minimal values in a particular application.

3.1.2 Interface Requirements

The FP interfaces with applications (which use the Application Interface), the Virtual Terminal, and the IISS environment by sending and receiving NTM messages. The interface to the NTM is provided by the User Interface Monitor (UIM) which is responsible for interpreting received messages and processing them accordingly.

3.1.2.1 Interface Block Diagram

The structure of the FP interfaces is shown in Figure 3-1.

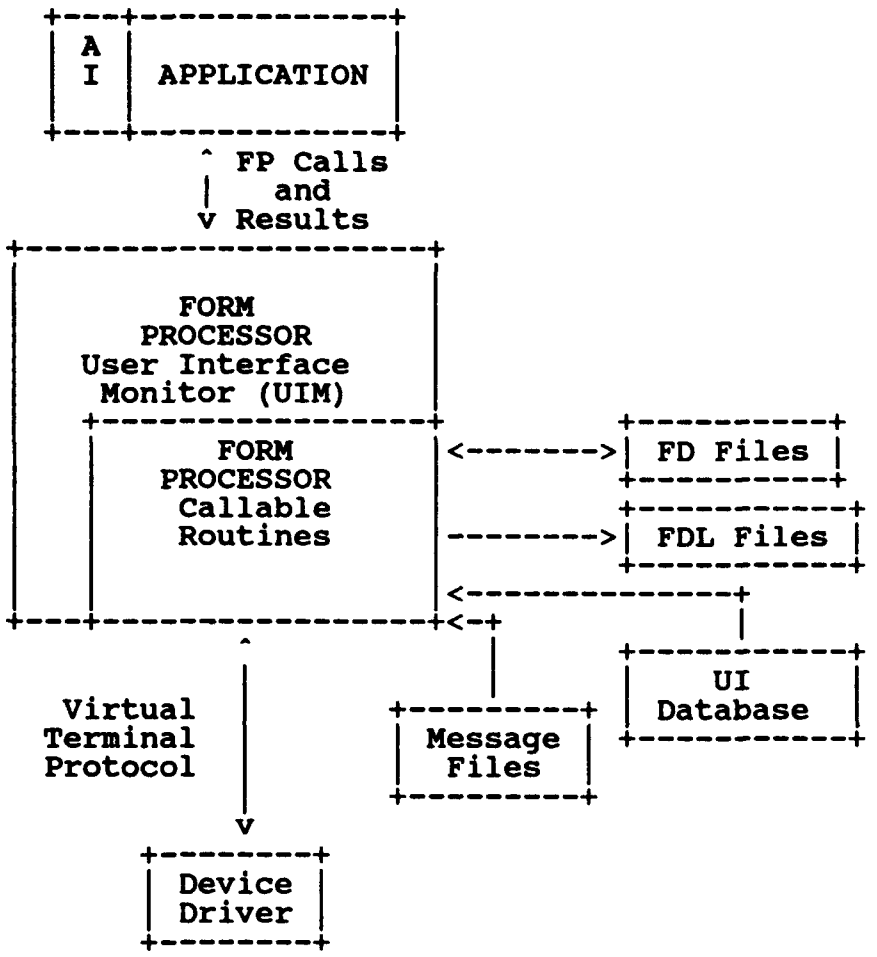


Figure 3-1 Form Processor Interfaces

3.1.2.2 Detailed Interface Definition

3.1.2.2.1 Application

The Form Processor interface for IISS applications is the set of callable routines defined in the IISS Form Processor User Manual[3]. The Application Interface intercepts these calls and converts them into NTM messages. These messages are received by the UIM and converted back into the original FP routine calls.

3.1.2.2.2 Device Drivers

The Form Processor interfaces with Device Drivers through a Virtual Terminal Protocol which is documented in the Virtual Terminal Development specification.

3.1.2.2.3 Data Files

The Form Processor interfaces with several data files. These are message files, FDL files and FD files. The message files are produced by the MM application and the format is described in Section 3.5.1.2. The FDL files are output when forms are created and modified by an application at execution time. The structure of these files is described in [9] the Forms Language Compiler Development Specification. The FD files are generated when FDL files are compiled by the Forms Language Compiler. The structure of these files is described in Section 3.5.1.1.

3.2 Detailed Functional Requirements

The following sections describe the detailed functional requirements of the Form Processor. The Form Processor includes a Graphics Support System based on the PHIGS Draft Standard. Terminology for both the text-based Forms Processor and PHIGS will be maintained in this implementation. Where these concepts are similar or equal they will be noted. The proposed C bindings routine names for the functions are enclosed in parenthesis following the reference to the functions.

In PHIGS Structures are very similar in concept to "forms". Structures provide, the vehicle for traversing the hierarchy. The structures can execute other structures providing a structure within a structure capability.

The PHIGS structures become active when they are "Posted" to a workstation. In the From Processor, this is comparable to adding a form to a window. The concept of a window in the Form Processor is consistent with the workstation in PHIGS.

3.2.1 Form Processing

Forms consist of rectangular areas on the terminal screen or hardcopy device. These areas are called fields and allow the user to enter and view variable data. There are nine types of form fields:

- o Items
- o Forms
- o Windows
- o Graphs
- o Polymarkers
- o Polylines
- o Text
- o Fill Areas
- o Cell Arrays

In ITEM is a field that holds a specific piece of data. Either the application program or the user can fill in the data value.

As the number of forms and item fields per form increase, there may be groups of items that are shared by more than one form or that make sense by themselves. These logical groups of

items can be made into separate forms and incorporated into others as a unit by defining an area on the host form as a FORM field. Forms can be nested to any level in this manner.

WINDOW fields provided the capability of changing part of a form at run time. The WINDOW is used as a place holder on the form. At run time, the application determines what forms the window will contain. A WINDOW field can contain more than one form at a time although only one form will be visible. A WINDOW can contain a WORKSTATION. Each form added to the window creates a page and just as with a book, only the current page can be seen.

WORKSTATIONS are representations of two or three dimensional objects. A WORKSTATION is analogous to a FORM.

GRAPHS are pictures correlated with data that alters as the data changes. GRAPHS are dynamic (not predefined) pictures. A GRAPH may be displayed upon a FORM or a WINDOW.

POLYMARKERS are symbols (such as a cross or a dot) which may appear in a FORM.

POLYLINES are geometric primitives consisting of one or more connected line segments. They may appear in FORMs.

TEXT are character strings which may appear in two ways; as characters which are displayed in planes (i.e. a perspective view), or as 2-D annotations which are not part of any object.

FILL AREAS are geometric primitives consisting of planar areas which are filled with colors or patterns.

CELL ARRAYS are geometric primitives which consist of a rectangular grid of equal size rectangular cells each with an assigned color or shade.

3.2.1.1 Open List

The Open List is an internal Form Processor list that contains all forms that the application has opened for use along with information on where the form is used.

3.2.1.2 Display List

The Display List is an internal Form Processor list that contains only those forms that have been added to the screen and are currently displayed on the screen, along with information on where those forms are used.

In summary, windows can contain forms, workstations, and graphs. Forms can contain items, forms, windows, polymarkers, polylines, text, fill areas, and graphs. To display forms, an application builds a Display List similar to that used in many high-powered graphics terminals. The Display List begins with the form PSCREN (the system form) which contains the window SCREEN which is the root for all user forms. Since windows contain forms which in turn can contain windows, items

and forms, the Display List could be a forms hierarchy like that shown in Figure 3-2. Before a form can be put on the Display List, it must be put on the Open List.

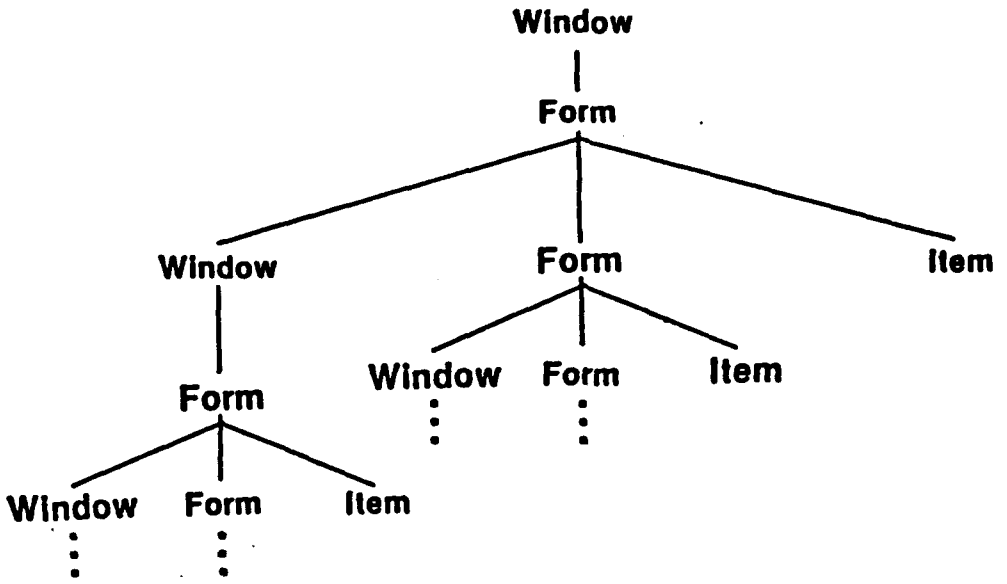


Figure 3-2 Sample Display List

The Display List allows users to easily transfer from form to form or have forms within other high level forms. It also provides the basis for running interactive programs concurrently.

Fields on the Display List are identified using qualified names. A fully qualified name specifies the name of each form, window, and item on the path from a specified starting point to the field being identified. The name of an array element is the name of the array followed by subscripts enclosed in parentheses (e.g., "array(2)"). The name of a page in a window is the name of the window followed by a page number enclosed in angle brackets (e.g., "window<1>"). Names are separated by periods and the qualified name ends with a semicolon (e.g., "window<1>.form.item(2);"). A qualified name need not be fully qualified provided the omitted elements are not necessary to identify different fields with the same name.

A qualified name that begins with a period is an absolute name which means the path starts at the top of the Display List (i.e., PSCREN). A qualified name that does not begin with a period is a relative name which means the path starts at the current default field. The default field is initially set to PSCREN but may be changed.

Fields can depend on other fields for their locations and/or values. Each field contains lists of other fields that depend on its horizontal location, vertical location and value.

Implementing the User Interface forms based style of interaction includes the following form processing activities:

- Controlling the Form Processor
- Opening and closing forms
- Creating and modifying forms
- Adding and removing forms from the current display
- Transferring data between forms and applications
- Displaying forms and processing user input
- Getting and setting the cursor position
- Displaying messages
- Creating and modifying logical devices

These activities are performed by callable routines that operate on a network of statically and dynamically allocated data structures.

The data structures for text-based forms and as graphics primitives on forms are:

UID - allows access to the network.

USR - contains information about each logged on user.

PD - contains information about each physical device.

AP - contains information about each running application.

FPD - represents information about logical devices.

FIELD - represents forms.

RELPOS - represents a relative position.

FORM - points to all the displayed instances of the form and a list of attributes for the form.

ITEM - points to the help message for the item and a data buffer which contains the current and previous values of the item.

PROMPT - points to the prompt text.

ATTMAP - represents background and display attributes.

MSGBUF - represent messages.

ENODE - represents field values.

COORD - represents a list of horizontal and vertical positions which may or may not be relative

The data structures for Graphs are:

AXIS - points to an axis definition for a graph form.

PLOT - contains information common to pie, bar, and line graphs.

LEGEND - contains legend information for a graph form.

SEGINFO - contains information about a pie segment.

LABEL - contains information for graphics text string.

IDAT - contains independent data information for a curve.

DDAT - contains dependent data information for a curve. A total curve definition for either a bar or line graph consists of the DDAT node along with the parent IDAT vertex.

The data structures for 3-D Graphics are:

This is PHIGS data which needs to be incorporated into the Forms data structures.

PHIGS Description Table -
contains information on the number of available workstation type, number of open workstations, available character sets.

PHIGS Traversal State List -
contains information on text font, attributes, and index values for Polylines, Polymarkers, etc.

PHIGS State List - contains information on the set of open workstations, name of open structure, current element pointer, and edit mode.

PHIGS Workstation State List - contains information on the open workstation, workstation identifier,connection identifier, and workstation type.

PHIGS Workstation Description Table - contains information on the type of workstation , whether output,input, metafile, etc.

Sections 3.5.3 and 3.5.4 contain the detailed definitions of these structures. The following sections describe how these structures are used by the callable routines to perform the form processing activities.

3.2.1.3 Controlling the Form Processor

The routine INITFP must be called by an application before any other FP routines are called to perform initialization activities such as initializing the internal data structures.

The routine TERMFP must be called when all form processing is completed to allow the Form Processor to perform necessary cleanup activities such as freeing internal data structures.

The routine SETDQN sets the default field in the FPD structure, and the routine GETDQN returns the absolute fully qualified name of the current default field.

The routine REDRAW ALL STRUCTURES (PREDRAWALLSTRUCT) will execute all deferred actions, and will cause all structures posted to this workstation to be redisplayed.

The routine UPDATE WORKSTATION (PUPDATEWS) will execute all deferred actions, and will depending on flag status, either cause updated blocks to be displayed or, cause all structures posted to this workstation to be redisplayed.

The routine, INQUIRE LIST OF AVAILABLE WORKSTATION TYPES, returns the list of workstation types which can be used as parameters to popenws.

The routine, INQUIRE WORKSTATION CONNECTION AND TYPE, returns the connection identifier and specific workstation type associated with the specified open workstation.

The routine, PARFQN, lets you find out the name of the field and its type at a specified level of a specified fully qualified name.

The routine, EVLINT, evaluates an integer value expression.

The routine, EVLREA, evaluates a real value expression.

The routine, EVLSTR, evaluates a string value expression.

The routine, GPAGE, returns the name of the form that a specified page of a window contains.

This routine, GWINDO, returns the number of pages in a window.

The routine, INQDEV, returns the width and depth of a physical device, and whether or not the device supports graphics.

The routine, Inquire Display Space Size, returns the display size of the specified workstation.

The routine, Inquire Display Space Size 3, returns the display size of the specified workstation.

3.2.1.4 Opening and Closing Forms

The routine OPNFRM reads a form definition file and creates the corresponding FIELD structures which are then added to the Open List. The routine CLSFRM releases the FIELD structures on the Open List when that form is no longer required for processing.

3.2.1.5 Creating and Modifying Forms

The routines described in this section enable application programs to create new forms, add or remove fields from forms, change form and field characteristics, and store forms at execution time. Both the Open and Display Lists are modified by the routines that perform set and remove operations. Only the Open List is modified by the routines that perform create and save operations.

3.2.1.5.1 Creating and Saving Forms

The routine CRTFRM creates a FIELD structure for a form which may then be further defined using other routines to include such things as prompts, fields, or particular attributes. The form size defaults to 0 0 meaning it is not a fixed size and the attribute is transparent. The routine MAKFRM creates a FIELD structure for a form with a specified size and attribute. The form may then be further defined using other routines as described for CRTFRM. The routine SAVFRM saves the FIELD structures for a form as an FDL source file, a compiled form definition, or both.

The routine REPFRM creates a copy of the FIELD structures for a form with a new form name. All characteristics of the original form are replicated in the new form at the time of the copy operation. After the copy, the two forms are entirely separate entities.

The routine CPFORM copies a form on the display list to a new instance of the form on the display list. The data from the copied form is replicated to the new form.

3.2.1.5.2 Adding or Removing Fields

The routines described in this section modify the Open List and then make the same changes on the Display List by following the next use pointers.

Text-Based Forms

The routine CRTFLD creates a new FIELD structure on a form. Required field characteristics have default values. The field is a non-displayed ITEM. This means that the location must be specified using the routine SETLOC in order for the item to appear on the screen. The size of the field defaults to one and its display attribute is text. Other routines may be used to change such things as prompts, help and array dimensions. The routine ADDFLD also creates a new FIELD structure on a form but allows the location, size, field type and display attribute to be specified without calling additional routines.

The routine REPFLD creates a copy of a FIELD structure with a new name. The form containing the field copy may be the form containing the original field or a different form. All characteristics of the original field are replicated in the new field at the time of the copy operation except the location. The field copy will not have a location and will not appear when its containing form is displayed unless another routine is used to define a valid location for it. The routine MVRFLD creates a copy of a FIELD structure with a new name and a specified location. After the copy using either routine, the two fields are entirely separate entities.

The routine RMVFLD removes an existing FIELD structure from a form. If other locations depend on the field being removed, an error is returned. If other field values depend on the field, the values become undefined (i.e., the next time the value is calculated, *'s will be displayed in the field). The routine GFMFLD returns the fields that are contained on a specified form.

The routine GDPFEX returns the fields whose values depend on a specified field. The routine GDPFLC returns the fields whose locations depend on a specified field.

3-D Graphics Forms

The routine POLYLINE (PPOLYLINE) will depending on the edit mode, insert a Polyline element into the open structure. During a structural traversal a connected sequence of straight lines is generated starting from the first point and ending at the last point. The current values of the Polyline attributes are bound to the parameter.

The routine POLYMARKER (PPOLYMARKER) will depending on the edit mode, inset a Polymarker element into the open structure. During structure traversal a sequence of markers is generated to

identify all the given positions. The current values of the marker attributes as defined in the PHIGS traversal state list are bound to the parameter.

The routine TEXT (PTEXT) will during structure traversal, generate a character string. The local coordinate system is the x y plane as defined in the modeling coordinate system. The current values of the Text attributes are bound to the parameter.

The routine FILL AREA (PFILLAREA) will during structure traversal, generate an implicitly closed polygonal area. This function specifies the three dimensional form of the fill area parameter. The current values of the fill area (interior) attributes is defined in the PHIGS traversal state list are bound to the parameter.

The routine CELL ARRAY (PCELLARRARY) specifies the form of the cell array parameter. During structure traversal a cell array is drawn using the cell rectangle corners, the number of cells along each axis and the color index array.

The routine EXECUTE STRUCTURE (PEXECUTESTRUCT) will during a structure traversal, execute a specific structure. During structure traversal the following actions shall occur. Traversal of the current structure will be suspended, the current state of the PHIGS traversal state list is saved, the execute structure network is completely traversed, the saved PHIGS transversal state list values are restored, and the transversal of the current structure is resumed.

The routine OPEN STRUCTURE will create the specified structure if necessary and open it.

The CLOSE STRUCTURE routine will close the current open structure.

The DELETE ALL STRUCTURE will delete all existing structures (as if by PDELSTRUCT).

The routine DELETE STRUCTURE, unposts the specified structure from all workstations and deletes the specified structure and all references to it. If the specified structure is currently open, it is closed prior to deletion and reopened afterwards.

The routine, DELETE STRUCTURE NETWORK deletes (as if by pdelstruct) the specified structure and all structure referenced by it whether directly or indirectly except that if ref_flag is KEEP, referenced structures which are also referenced by structures outside of the network are not deleted.

Graph Forms

The routine GRALOC specifies the location of a graph with respect to its containing form. This routine affects the display list only. Inputs are qualified name, qualified name of the vertical reference, vertical point on the reference (1=top, 2=center, 3=bottom), vertical reference (1=top, 2=center, 3=bottom), point on the graph, vertical distance from reference field to graph reference point, qualified name of horizontal reference, horizontal point on the reference field (1=top, 2=center, 3=bottom), horizontal reference point on the graph (1=top, 2=center, 3=bottom), and horizontal distance from reference field to graph reference point. The only output is a return code.

The routine DELWHR deletes a "where data is located" expression on a graph. The routine effects the display list only. The qualified name of the graph or the curve is input. A return code is the output.

The routine ADDWHR adds a "where data is located" expression to a dataset. If the clause is to be a list, the routine can be called more than once. Only the display list is effected by the routine. Inputs are the qualified name of the graph or curve, the qualified name of the field containing the where clause, and the type (independent or dependent). A return code is the only output.

The routine ADDIAX specifies the axis that is to be used for the independent axis. If the graph is a pie chart, an error is returned. Only the display list is effected by the routine. The qualified name of the graph and the name of the independent axis are inputs. A return code is the output.

The routine PGDATA provides data to the Form Processor in lieu of a "where data is located" clause. If the data location clause is already present, an error is returned and the clause must be removed. Inputs are the qualified name of the graph, the number of data points that are being passed in by the following argument, the data that is to be plotted and the type (independent or dependent). A return code is the output.

The routine DELBUN deletes a graphics attribute bundle. If the name of the bundle is omitted, all bundled attributes are deleted. This routine effects both the open and display lists. Inputs are the qualified name of the graphics field and the name of the attribute bundle. The only output is a return code.

The routine DEFBUN defines a graphics attribute bundle. This routine effects the open and display lists. Inputs are the qualified name of the graph, the name of the attribute bundle, the length of the attribute string, and the list of attributes for the bundle being defined. A return code is the output.

The routine DELTLA deletes the tick mark labels from a particular axis instance on the display list. The qualified name of the axis and the type of labels (major or minor) are inputs. A return code is the output.

The routine DFMXMN defines the maximum and minimum values on an axis instance on the display list. If the values are set,the FP will determine the appropriate values based on the data. Inputs are the qualified name of the axis, the maximum value on the axis, and the minimum value on the axis. A return code is the output.

The routine ADDTLA adds tick mark labels to an axis instance on the display list. Multiple labels may be added by repeating the routine call. inputs are the qualified name of the axis, the type of label (major or minor), the length of the tick mark label string and the tick mark label string. The output is a return code.

The routine DEFWIN defines a graph window. The limits of the window allow the data coordinates to be correlated to the screen coordinates. If the maximum and minimum coordinates of the window are not specified, the default for business graphs will be the maximum and minimum data values adjusted for a 10% margin. Only the display list is effected by this routine. Inputs are the qualified name of the graph, the maximum horizontal value, the minimum horizontal value, the maximum vertical value and the minimum vertical value. A return code is the output.

The routine DEFGVU defines the extent that the graph is to fill the graph window (the range is 0 to 1). Only the display list is effected by this routine. Inputs are the qualified name of the graph, the maximum horizontal value, the minimum horizontal value, the maximum vertical value and the minimum vertical value. A return code is the output.

The routine ADTICS adds tick marks to a particular axis instance on the display list. Inputs are the qualified name of the axis, the name of the graphics attribute bundle to be used to display the tick mark labels, an indicator to determine whether the following value is a value step between ticks or an absolute number of ticks, or value of ticks. The output is a return code.

The routine DEFSEG creates or redefines a segment of a pie chart. This routine affects the display list only. Its input consists of a qualified name, a segment number, an explosion factor, fill color, pattern name and alternate pattern name. The only output is a return code.

The explosion factor is the percentage of the radius to project the segment from the center of the circle. For example, an explosion factor of 2 will project the segment a factor of (radius + (0.02 * radius)) from the circle. Fill color is the color with which to fill the segment. The available colors are BLACK, WHITE, MAGENTA, BLUE, RED, CYAN, YELLOW, GREEN.

Pattern name is the name of the pattern with which to fill the segment. Alternate pattern name is the name of the pattern to use if the physical device is monochromatic.

The routine PERQUA adds a percent or quantity definition to a segment. This routine affects the display list only. Inputs include the qualified name, the segment number, the percent or quantity, and an inside or outside flag. The only output is a return code.

The routine LABSEG adds a label to a pie segment. A segment may have more than one label by calling this routine the desired number of times. The routine affects the display list only. Inputs are qualified name, segment number, length of the segment label, and the segment label. the only output parameter is a return code.

The routine LEGLAB adds a legend label for a curve or segment. Any number of legend labels may be added for a entity by calling the routine repeatedly. The routine affects the display list only. Inputs are qualified name, legend type (curve or segment), segment number (if type is segment), curve name (if type is curve), length of the legend string, and the name of the attribute bundle that will be used to display the legend string. The only output is a return code.

The routine DELSEG deletes a pie segment. The data will also be deleted, and this will affect the drawing of the other segments in a pie chart since the total value of the pie will be altered. This routine affects the display list only. Inputs are qualified name and segment number. A return code is the only output.

The routine DEFGRA defines a graph. This routine affects the open and display lists. Inputs are graph name, graph type and the name of the attribute bundle the is used for the background of the graph. A return code is the output parameter.

The routine ADDLEG adds a legend definition to a graph. The legend labels are added separately. The routine affects the display list only. Inputs are the qualified name, a flag to indicate whether or not the legend should be enclosed by a box, the qualified name of the vertical reference point, the vertical point on the reference field (1=top, 2=center, 3=bottom), vertical reference point on the graph (1=top, 2=center, 3=bottom), vertical distance from the reference field to the graph reference point, qualified name of the horizontal reference, horizontal point on the reference field (1=top, 2=center, 3=bottom), horizontal reference point on the graph (1=top, 2=center, 3=bottom), and horizontal distance from reference field to graph reference point. The output is a return code.

The routine ADGLAB adds a label to a graph. A graph may have any number of labels by calling the routine repeatedly. The routine effects the open and display list. Inputs are graph name, length of the label string, the label string, the name of the attribute bundle used to display the label, the qualified name of the vertical reference point, the vertical point on the reference field (1=top, 2=center, 3=bottom), vertical reference point on the graph (1=top, 2=center, 3=bottom), vertical distance from the reference field to the graph reference point, qualified name of the horizontal reference, horizontal point on the reference field (1=top, 2=center, 3=bottom), horizontal reference point on the graph (1=top, 2=center, 3=bottom), and horizontal distance from reference field to graph reference point. The output is a return code.

The routine DELLEG deletes a legend from a graph. The routine effects the display list only. Input is the qualified name of the graph and the output is a return code.

The routine DELGLA removes all the graph labels. The labels on the axes or segments are not effected. This routine effects the display and the open lists. Input is the qualified name of the graph and the output is a return code.

The routine DEFCRV defines a curve for a bar or linear plot. This routine effects the display list only. Inputs are the qualified name of the graph, the name of the curve, the name of the dependent axis, the name of the graphics attribute bundle that the curve itself is to be displayed with, the name of the alternate graphics bundle that is to be used for display when the physical device is monochromatic, the name of the fill color (BLACK, WHITE, MAGENTA, YELLOW, RED, BLUE, CYAN, or GREEN), the name of the pattern that is to used to fill the area under the curve, the name of the alternate fill pattern to use if the physical device is monochromatic , type (additive or absolute) and the name of the curve that is being added to. A return code is the only output parameter.

The routine DELCRV deletes a curve from the display list. The qualified name of the curve is input and a return code is output.

The routine DELSLA deletes the labels from a pie segment. This routine effects the display list only. The qualified name of the axis and the segment number are inputs. A return code is the output.

The routine DELALA deletes all the labels from an axis instance on the display list. The input is the qualified name of the axis and the output is the return code.

The routine DELAXS deletes an axis definition from both the open and display lists. Inputs are the qualified name of the graph and the axis name. The output is a return code.

The routine DFAXLC determines where the axis is to be placed. This routine effects the display list only. Inputs are the qualified name of the axis, the qualified name of the vertical reference point, the vertical point on the reference field (1=top, 2=center, 3=bottom), vertical reference point on the graph (1=top, 2=center, 3=bottom), vertical distance from the reference field to the graph reference point, qualified name of the horizontal reference, horizontal point on the reference field (1=top, 2=center, 3=bottom), horizontal reference point on the graph (1=top, 2=center, 3=bottom), and horizontal distance from reference field to graph reference point. The output is a return code.

The routine ADDALA adds axis labels to an axis instance on the display list. Multiple labels may be added by repeating the routine call. The qualified name of the axis, the length of the axis string and the label string are inputs. A return code is the only output.

The routine DEFAXS defines an axis. Both the display and open lists are effected. Inputs are the qualified name of the graph the axis is linked to, the axis name, the direction, the scale (linear or logarithmic), the grid, the length of the axis, and the name of the attribute bundle to be used for display purposes. A return code is the output.

3.2.1.5.3 Changing Characteristics

Any form and field characteristics that can be specified using the Forms Definition Language can be changed using the routines described in this section.

Text-Based Forms

The routine SETATT sets the ATTDEF structure in an ATTMAP structure attached to a form FIELD structure. The ATTMAP structure is created if it doesn't already exist. If the primitive conflicts with an existing primitive, the existing primitive is removed. The routine RMVATT removes either a specific attribute definition or all attribute definitions from a form. The routine INQATT returns the attributes which are defined for a form or the primitives that are contained in a specific attribute definition.

The routine SETDIS sets the permanent attribute of a FIELD structure. The routine INQDIS returns the current permanent attribute of a FIELD structure.

The routine SETHLP sets the help string of an item FIELD structure. The routine INQHLP returns the current help string of an item FIELD structure. The routine RMVHLP deletes the current help string of an item FIELD structure.

The routine SETSIZ sets the size of a FIELD structure. The routine INQSIZ returns the size of a FIELD structure.

The routine ADDRPT creates a new array FIELD structure which contains a specified number of the previously existing FIELD structures. The routine RMVDIM eliminates an array FIELD structure.

The routine SETRPT sets the members of the ARRAY structure in an array FIELD structure. The specified FIELD structure must exist. The RMVRPT removes all of the array FIELD structures associated with a FIELD structure, resulting in the FIELD structure being attached directly to its parent form FIELD structure. The INQRPT returns the members of an ARRAY structure.

The routine SETVAL parses the supplied value expression into a tree of ENODE structures which is then attached to the specified FIELD structure (replacing any previous value). The routine INQVAL traverses the ENODE structure tree to regenerate the value expression which it returns. The routine RMVVAL deletes the ENODE structure tree.

The routine SETAPR parses the supplied appears if expression into a tree of ENODE structures which is then attached to the specified FIELD structure (replacing any previous value). The routine INQAPR traverses the ENODE structure to regenerate the appears expression which it returns. The routine RMVAPR deletes the ENODE structure tree.

The routine SETDOM sets a domain field of the ITEM structure in a FIELD structure. If the new setting conflicts with the settings of other domain fields, the other fields are adjusted to conform to the new setting (e.g., if justl is set and SETDOM is called to set justr, justl will be cleared). The routine RMVDOM clears all of the domain fields. The routine INQDOM returns the value of a domain field or fields.

The routine SETNAM changes the name in a FIELD structure. An error is returned if there are references to the field (e.g., relative locations or calculated values).

The routine SETLOC sets the location in a FIELD structure. The horizontal and vertical components of the location are specified separately. Each component consists of a reference point on the field being positioned, a field to position it relative to, a reference point on the related field, and the distance between the two reference points. If the related field is not specified, the second reference point defaults to the upper left corner of the containing form. Setting the location

of a FIELD structure to 0 0 makes it non-displayed. The routine INQLOC returns the location as specified of a FIELD structure. The routine INQABS returns the absolute row and column location of a FIELD structure relative to its containing form after resolving relative positions.

The routine SETTYP changes the type of a field. Any previously defined characteristics which do not apply to the new field type are deleted. Any required characteristics not previously specified, will automatically be defined with appropriate defaults. The routine INQTYP returns the type of a field.

The routine, INQDSZ, returns the size and depth of a particular instance of a field on the display list.

The routine, SETSLN, redefines a non-graphics line field. If the field does not previously exist, one is created.

The routine SETPRO adds a prompt to a FIELD structure. The routine RMVPRO removes a specific prompt or all prompts from a FIELD structure. The routine INQPRO returns a prompt defined for a field.

3-D Graphics Forms

The routine SET POLYLINE INDEX (PSETLINEIND) will during structure traversal, set the current Polyline index entry in the PHIGS traversal list to the value specified by the parameter. This value is used when creating subsequent Polyline and Polyline three output primitives.

The routine SET POLYMARKER INDEX (PSETMARKERIND) will during structure traversal, set the current Polymarker index entry in the PHIGS traversal state list to the value specified by the parameter. This value is used when creating subsequent Polymarker and Polymarker three output primitives.

The routine SET TEXT INDEX (PSETTEXTIND) will during structure traversal, set the current Text index entry in the PHIGS traversal state list to the value specified by the parameter. This value is used when creating subsequent Text output primitives.

The routine SET LINE TYPE (PSETLINETYPE) will during structure traversal, set the current line type entry in the PHIGS state list to the value specified by the parameter. This value is used when creating subsequent Polyline output primitives. This value does not effect the display of subsequent Polylines output primitives created when the current line type entry PHIGS traversal state is bundled. Line type values which are legal are 1) solid line 2) dashed line 3) dotted line 4) dashed dotted line 5) reserve for registration, and less than zero are implementation dependent. During structure traversal if the specified line type is not available on a workstation, line type 1 is used on that workstation.

of a FIELD structure to 0 0 makes it non-displayed. The routine INQLOC returns the location as specified of a FIELD structure. The routine INQABS returns the absolute row and column location of a FIELD structure relative to its containing form after resolving relative positions.

The routine SETTYP changes the type of a field. Any previously defined characteristics which do not apply to the new field type are deleted. Any required characteristics not previously specified, will automatically be defined with appropriate defaults. The routine INQTYP returns the type of a field.

The routine, INQDSZ, returns the size and depth of a particular instance of a field on the display list.

The routine, SETSLN, redefines a non-graphics line field. If the field does not previously exist, one is created.

The routine SETPRO adds a prompt to a FIELD structure. The routine RMVPRO removes a specific prompt or all prompts from a FIELD structure. The routine INQPRO returns a prompt defined for a field.

3-D Graphics Forms

The routine SET POLYLINE INDEX (PSETLINEIND) will during structure traversal, set the current Polyline index entry in the PHIGS traversal list to the value specified by the parameter. This value is used when creating subsequent Polyline and Polyline three output primitives.

The routine SET POLYMARKER INDEX (PSETMARKERIND) will during structure traversal, set the current Polymarker index entry in the PHIGS traversal state list to the value specified by the parameter. This value is used when creating subsequent Polymarker and Polymarker three output primitives.

The routine SET TEXT INDEX (PSETTEXTIND) will during structure traversal, set the current Text index entry in the PHIGS traversal state list to the value specified by the parameter. This value is used when creating subsequent Text output primitives.

The routine SET LINE TYPE (PSETLINETYPE) will during structure traversal, set the current line type entry in the PHIGS state list to the value specified by the parameter. This value is used when creating subsequent Polyline output primitives. This value does not effect the display of subsequent Polylines output primitives created when the current line type entry PHIGS traversal state is bundled. Line type values which are legal are 1) solid line 2) dashed line 3) dotted line 4) dashed dotted line 5) reserve for registration, and less than zero are implementation dependent. During structure traversal if the specified line type is not available on a workstation, line type 1 is used on that workstation.

The routine, ROTATE X, generates the three dimensional transformation which performs the specified rotation about the X axis.

The routine, ROTATE Y,generates the three dimensional transformation which performs the specified rotation about the Y axis.

The routine, ROTATE Z, generates the three dimensional transformation which performs the specified rotation about the Z axis.

The routine, SCALE, generates the two dimensional transformation which performs the specified scaling.

The routine, SCALE 3, generates the three dimensional transformation which performs the specified scaling.

The routine, SET CHARACTER HEIGHT, inserts a set character height element into the currently open structure. During traversal, this element will set the character height entry in the traversal state list which affects subsequent text primitives.

The routine, SET GLOBAL TRANSFORMATION, inserts a two dimensional set global transformation element into the currently open structure. During traversal, this element will set the global transformation entry in the traversal state list which affects subsequent primitives.

The routine, SET GLOBAL TRANSFORMATION 3, inserts a three dimensional set global transformation element into the currently open structure. During traversal, this element will set the global transformation entry in the traversal state list which affects subsequent primitives.

The routine, SET INTERIOR COLOR INDEX, inserts a set interior color index element into the currently open structure. During traversal, this element will set the interior color index entry in the traversal state list which affects subsequent fill area primitives.

The routine, SET INTERIOR STYLE, inserts a set interior style element into the currently open structure. During traversal, this element will set the interior style entry in the traversal state list which affects subsequent fill area primitives.

The routine, SET LOCAL TRANSFORMATION, inserts a two dimensional set local transformation element into the currently open structure. During traversal, this element will modify the local transformation entry in the traversal state list by composing the specified transformation with the current entry in the specified manner which affects subsequent primitives.

The routine, SET LOCAL TRANSFORMATION 3, inserts a three dimensional set local transformation element into the currently open structure. During traversal, this element will modify the

local transformation entry in the traversal state list by composing the specified transformation with the current entry in the specified manner which affects subsequent primitives.

The routine SET TEXT COLOR INDEX (PSETTEXTCOLOURIND) will during structure traversal, set the current Text color index entry in the PHIGS state list to the value specified by the parameter. During structure traversal if the specified text color index is not available on the workstation, color index 1 is used on that workstation.

3.2.1.6 Modifying the Display List

The routine ADDFRM copies a form FIELD structure from the Open List to the Display List underneath a window FIELD structure. References to other fields in locations and values are resolved into pointers to the fields and the final locations and values are computed. Likewise, references to attributes are resolved into pointers to the attributes' definitions. If the form being added is not already on the Open List, it will be opened automatically. The added form creates a new page in the window which "covers up" any previous pages but does not remove them.

The routine RPLFRM replaces the form FIELD structure currently on a specified page in a window with a different form from the Open List. The routine RMVPAG removes a specified page from a window as well as any subsequently added pages, uncovering the previously added page. The routine GWINDO returns the number of pages in a window, and the routine GPAGE returns the name of the form that a specified page of a window contains.

The routine ADDELM adds an element to the end of an open-ended array (i.e., an array with no specified number of elements) by copying the prototype element FIELD structure from the Open List to the Display List underneath the array FIELD structure. The routine NUMELM returns the current number of elements in a specified open-ended array.

The routine, DELELM, deletes an element from an open-ended array.

The routine PUTATT sets the permanent or temporary attribute in item FIELD structure. The temporary attribute is in effect the next time the form is displayed only. The permanent attribute is en effect for all subsequent displays. If a form is specified instead of a specific item, then PUTATT is applied recursively to all of the fields contained in the form. If a window is specified, then PUTATT is applied recursively to all of the pages contained in the window. The routine GETATT returns the values for temporary or permanent display of an item.

The routine PUTBAK sets the permanent attribute of a form or window FIELD structure. Temporary attributes are not supported for forms or windows. The routine GETBAK returns the permanent attribute of a form or window field.

The routine APRFLD modifies the display flag for any type of FIELD structure.

The routine POST STRUCTURE (PPOSTSTRUCT) add the specified structure to the table of posted structures in the workstation state list of the specified workstation. If the specified structure does not exist, a new empty structure is created.

The routine UNPOST STRUCTURE (PUNPOSTSTRUCT) will unpost the specified structure from the specified workstation by removing the structure from the table of posted structures from the workstation state list of specified workstations. If the specified structure does not exist, no action takes place. Note that unposting a structure does not delete the structure.

The routine UNPOST ALL STRUCTURES (PUNPOSTALLSTRUCT) will unpost all structures from the specified workstation by removing all structures from the table of posted structures from the workstation state list of the specified workstation. Note that unposting any structure does not delete the structure.

The routine OPEN WORKSTATION (POPENWS) is called by the application. PHIGS will request that the operating system establish the specified connection for a workstation characterized in the workstation description table by the workstation type. The workstation state list is allocated and initialized. The workstation identifier is added to the set of open workstations in the PHIGS state list.

The routine CLOSE WORKSTATION (PCLOSEWS) executes an UPDATE WORKSTATION (PUPDATEWS) for the specified workstation. The workstation state list is released. The workstation identifier is deleted from the set of open workstations in the PHIGS state list and from the list of workstations to which posted in the structure state list. The input queue is flushed of all events from all devices on the workstation for which are being closed. The specific workstation description table, created when the workstation was opened, becomes unavailable and the workstation type value associated with this specific workstation description table becomes undefined. The connection to the workstation is released.

3.2.1.7 Transferring Data

The routine PDATA puts data into a field. If the field is an item, then data is copied into the data buffer pointed to by the item FIELD structure. If the field is a form, sequential chunks of data are placed in each field on the form. If the field is a fixed-size (i.e., not open-ended) array, sequential chunks of data are placed in each element of the array. All other field types are skipped over.

The routine GDATA returns data from a field on a form analogous to PDATA. Either the current data values or the values prior to the previous OISCR (section 3.2.1.5) may be retrieved. The routine GDATLN returns the length of the data contained in a field.

3.2.1.8 Displaying Forms

The routine OUTSCR updates the terminal screen with the current Display List and the current cursor location. The routine OISCR also updates the terminal screen, but it waits for user input before processing continues. The name of the window where input is to occur must be specified; all fields outside this window are guarded and not enterable. Input data is received from the Device Driver whenever a function key is pressed. The data associated with enterable items is stored in the buffer pointed to by the items' FIELD structures. The function key is processed as specified in Section 3.2.4. If the key is an Application mode key, it is returned to the application. Otherwise, OISCR waits for additional input data.

Every item defined for a form has a value associated with it. If the value depends on another field, it is referred to as a calculated field and its value is recomputed prior to updating the terminal screen when a field it depends on has been changed since the last time the screen was updated. The order in which field values are recomputed is undefined and a field is only recomputed once per display. Consequently, using one calculated field in another field's value is not supported.

The routine, RMVPAG, removes a specified page from a window as well as any subsequently added pages, uncovering the previously added page.

The routine, ROTWND, rotates a window a given times 90 degrees in the counter-clockwise direction.

3.2.1.9 Getting and Setting the Cursor Position

Icons may be thought of as graphical figures that may be picked. There are many ways to identify the location of a cursor or the result of a graphical pick. Since graphical figures may be incorporated into forms, one may use the FP routine GETCUR to retrieve the cursor location of the form. Similarly, the OUTCUR routine may be used to set the cursor position. Graphical figures may also be PHIGS structures. Therefore, some PHIGS location routines are included for completeness.

The routine GETCUR returns the fully qualified name of the field and the row and column within that field where the cursor was located at the end of the previous OISCR. The routine PARFQN breaks a qualified name into its component parts. The routine PUTLOC sets the output cursor position in the FPD structure to the specified row and column within a field. The routine PUTCUR is identical to PUTLOC but the row and column are are both assumed to be one rather than being specified in the call.

3.2.1.10 Displaying Messages

The routine PMSGLS displays a specified message to the user. The routine PMSGLC displays the message that corresponds to a specified return code. The MM application is used to

create the message definition files whose format is described in Section 3.5.1.2. Messages are displayed when the terminal screen is updated and are removed at the end of the next OISCR.

3.2.1.11 Creating and Modifying Logical Devices

The FPD structure corresponding to an application is called a logical device. By creating additional logical devices, an application can appear to the user as multiple applications.

The routine INQLDV returns the id of the current FPD structure. The routine OPNLDV creates a new FPD structure for an application. The routine CHGLDV makes the specified logical device the current Display List for all following Form Processor calls. The routine CLSLDV deletes the FPD structure associated with a logical device. Neither the current logical device nor the default logical device can be closed.

The routine GETLDV returns the size and location of the specified logical device. The routine SETLDV is used to set the size and location of the specified logical device. The routine MOVLDV moves a logical device to a different physical device.

These are program callable functions. Section 3.2.5.5 describes the Window Manager function keys which allow the terminal user to change the size and location of a logical device on the actual terminal screen.

The routine, CLRLDV, removes a logical device from the screen. The definition is removed only from the VT and not from the FPD list in the Form Processor.

The routine, SCRLDV, scrolls a logical device by modifing the row and column.

The routine OPEN PHIGS (POPENPHIGS) must be called by the application before any other graphics routines are called. The PHIGS state list is allocated and initialized.

The routine CLOSE PHIGS (PCLOSEPHIGS) must be called by the application on completion of graphics. The PHIGS state list and the workstation description tables become unavailable. All PHIGS related buffers are released and all PHIGS files are closed.

3.2.2 Maintaining User Profiles

When a user first connects to the IISS environment, a USR structure, a PD structure for the logon terminal, and an FPD structure to be used by the UI are created. The Form Processor displays a form where the user enters logon information (user id, password, and role) and validates this information against the User Interface (UI) database. If the entered logon information is not valid, an error message is displayed and the user is allowed to correct the information. If valid information is not entered after five attempts, the user is disconnected. When valid information is entered, the user name and role are stored in the USR structure. The user's templates for Form Definition (FD) file names and Form Definition Language

(FDL) source file names which were retrieved from the UI database while validating the logon information are also stored in the USR structure. The IISS Function Screen is then displayed.

The routine GTUINF returns the name and role of the user running the application from the USR structure. The routine GTUSYM returns the file name templates from the USR structure. It is intended for use only by components of the UI (e.g. FLAN) and should not be called by user written applications.

A user invokes a function by filling in the IISS Function Screen and pressing the <ENTER> key. If the user has changed his role, the new role is validated against the user's user id as defined in the UI database. If the role is not valid, an error message is displayed and no other processing occurs. If the role is valid or was not changed, the selected function, if any, is validated against the user's role as defined in the UI database. If the function is not valid, an error message is displayed. When a valid function is entered, the definition of the function in the UI database is checked. It it is an internal function (an integral part of the UI), it is simply called.

If the specified function is a remote application (user written), a startup message (type UM) specified in the function definition is sent to the application. The list of PD structures associated with the user is searched for the device name and device type specified on the IISS Function Screen. If a matching PD structure is not found, one is created and a device startup message is sent to start the device. An AP structure associated with the user is created for the application and an FPD structure is also created for the application and associated with the AP and PD structures.

3.2.3 Virtual Terminal Pass-through

Virtual Terminal (VT) pass-through allows an application to use the Virtual Terminal protocol to communicate directly with the Device Driver. The routine INITVT enables the VT pass-through mode. While in this mode, the application talks directly to the terminal and no form processing routines may be used. The routine PUTVTI sends a buffer of VT commands to the Device Driver. The routine GETVTI returns a buffer of VT commands generated by the Device Driver in response to user input. The routine TERMVT disables VT pass-through mode allowing form processing routines to be used again.

3.2.4 NTM Message Processing

The Form Processor processes three kinds of messages:

- Device driver messages
- Application messages
- IISS environment messages
- PHIGs support messages

3.2.4.1 Device Driver Messages

Device driver messages control devices, communicate their status, and transfer data.

A device startup message (type DE) is sent to initiate a new slave device. A shutdown message (type SD) is sent to terminate a device. Receiving a device startup message (type DE) indicates a new user connection and results in logon processing (Section 3.2.2).

A device intitated message (type DI) is received from a slave device and contains device specific information. A device error message (type ER) is received when a device driver encounters an application specific error and causes the application to be terminated. A device abort message (type AB) is received when a device driver encounters a fatal error and causes all applications on the device to be terminated. If this device is the user's logon device, all of the user's applications are terminated and the user is logged off. A device acknowledgement message (type DA) is received when the device driver has finished processing commands and an acknowledgement was requested by sending a device query message (type DQ).

Device data messages (type DD) are received as a result of user input. The data is used to update the current display and then function key processing is performed. Device data messages are sent to update the terminal screen.

3.2.4.2 Application Messages

An application message (type AI) is received from the an Application Interface routine and is processed by calling the corresponding FP routine. An application message containing the results of the call is then sent back to the Application Interface routine.

3.2.4.3 IISS Environment Messages

Shutdown pending messages (type SP) are received from the time a shutdown is scheduled until it actually occurs. The time remaining until actual shutdown is displayed as a message to all users. A cancel shutdown message (type CS) is received when a shutdown is cancelled and is also displayed to all users. A shutdown message (type SD) is received when the actual shutdown occurs. All applications are terminated and all users are logged off.

3.2.4.4 PHIGs Support Messages

Messages which may be as a result of an operation error from PHIGS will displayed on the form.

3.2.5 Function Key Processing

Function keys are used to perform user controllable versus program controllable Form Processor and application functions. To allow for a large number of functions, modes are defined.

The function performed by a key depends on the current mode setting which is displayed in the message line of the terminal screen. Some keys, called control keys, perform the same function in all modes. The following sections describe these keys and the keys available in each mode.

Additionally, mouse buttons may be used as function keys if supported by the application mode.

3.2.5.1 Control Keys

The <ENTER> and <QUIT> keys are normally processed as Application mode keys, but they are also used by the Form Processor functions as described in this and following sections.

The <MODE> key is used to cycle through the available modes.

The <HELP> key is used to invoke help for an input item. If the item containing the cursor has a help message defined, the message is displayed. If it has a help form defined, the form is added to the window SCREEN and is removed when the <QUIT> key is pressed. If it is defined as having application help, any currently displayed help forms are removed and the <HELP> key is processed as an Application mode key as described in Section 3.2.5.2. If there is no help defined for the item or the cursor is not in an item, a "no help available" message is displayed.

There is a message queue associated with each application the user is running. Any of these messages can be displayed in the message line on the terminal screen. The most recent message is displayed by default and a different message can be selected by changing the message number. The <MESSAGE QUEUE> key displays a form containing all of the messages for the application containing the cursor.

3.2.5.2 Application Mode Keys

Application mode keys normally operate as defined by the application, but the <ENTER> and <QUIT> keys are also used by Form Processor functions as noted in the previous section. When an application mode key is pressed, the FP canonicalizes the user entered data (e.g., justification, case conversion) and, except when the <QUIT> key is pressed, validates the data on the form. If validation errors occur, the offending fields are highlighted and the user is given the opportunity to correct them.

3.2.5.3 Scroll/Page Mode Keys

The definition of an array of fields can indicate that fewer elements are to be displayed than actually exist. Scrolling may only take place if the array elements are homogeneous (e.g., in an array of windows, each window must contain the same form). When a user asks for scrolling in a specified direction (either horizontally or vertically), the Form Processor searches backwards along the path to the field containing the cursor for the first field which may be scrolled

in the requested direction. This field is then scrolled by one element in the requested direction. Error messages are displayed if no field is found or the field is already scrolled as far as possible.

Paging is identical to scrolling by the number of elements displayed on the screen.

The Scroll/Page mode function keys are:

<SCROLL UP>
<SCROLL DOWN>
<SCROLL LEFT>
<SCROLL RIGHT>
<PAGE UP>
<PAGE DOWN>
<PAGE LEFT>
<PAGE RIGHT>

If the field is defined as application scrolled, the Scroll/Page mode keys are processed as Application mode keys.

3.2.5.4 Text Editor Mode Keys

The Text Editor mode function keys move, copy, delete and substitute text among item fields. These function keys are:

<SEARCH>	Makes a forward or backward search for the first occurrence of the specified string.
<SEARCH NEXT>	Continues a search for the next occurrence of the previously specified string.
<REPLACE>	Replaces the first occurrence of a specified search string with a specified new string.
<REPLACE NEXT>	Replace the next occurrence of the specified search string with the previously specified new string.
<INSERT LINE>	Inserts a blank line in the text.
<DELETE LINE>	Deletes a line of text from the current cursor position to the end of the line.
<PASTE>	Pastes previously deleted into the current buffer. The format of the text is maintained.
<FILL>	Pastes previously deleted text into the current buffer reformatting the text according to the current fill margins.
<MIDLINE BREAK>	Breaks a line of text and moves it to the next line.
<DELETE LINE>	Removes all the text from an item.
<RESTORE>	Replaces the original text in an item that has been edited.

<REPEAT> Specifies a number of times to perform the action of the next TEXT EDITOR mode function that is pressed.

<FILL MARGINS> Sets up margins to be used when the fill function is performed.

3.2.5.5 Window Manager Mode Keys

In using the Display List and form hierarchy concepts, windows become stacked on forms and logical devices become stacked on the terminal screen. The Window Manager mode keys are used to manipulate the stacks of windows and logical devices. These function keys are:

<SELECT> Makes a window on the current display the top window in the stack so that its size and location can be changed and the information in it can be scrolled. It also allows the window to be viewed completely.

<RESTORE> Returns the selected window to its previous position in the stack.

<SIZE> Makes the selected window larger or smaller.

<LOCATION> Moves the selected window to a new location on the screen.

<SCROLL UP> Scrolls the form which is displayed in the selected window up.

<SCROLL DOWN> Scrolls the form which is displayed in the selected window down.

<SCROLL LEFT> Scrolls the form which is displayed in the selected window to the left.

<SCROLL RIGHT> Scrolls the form which is displayed in the selected window to the right.

<HOME VIEW> Returns a form that has previously been scrolled to its original position in the window.

<FUNCTION> Displays the IISS Function Screen so another application can be started.

<APPLICATION> Makes an application the top application in the stack so that you can view and manipulate its initial window.

These functions can also be performed using the information displayed on the Application Status Form (see section 3.2.4.6).

3.2.5.6 Status Mode Keys

The Status mode keys are used to display and modify the status of the applications a user is currently running.

The <DEBUG> key toggles debug mode ON and OFF. When debug mode is ON, system status and error messages are displayed in the message line on the terminal screen and recorded in the system message queue.

The <APSTAT> key displays the Application Status Form shown in Figure 3-3. This form displays information about the status of the applications (logical devices) on the terminal screen. All the functions that can be performed using the Window Manager mode function keys can also be performed from this form. Applications can also be aborted or moved to another physical device.

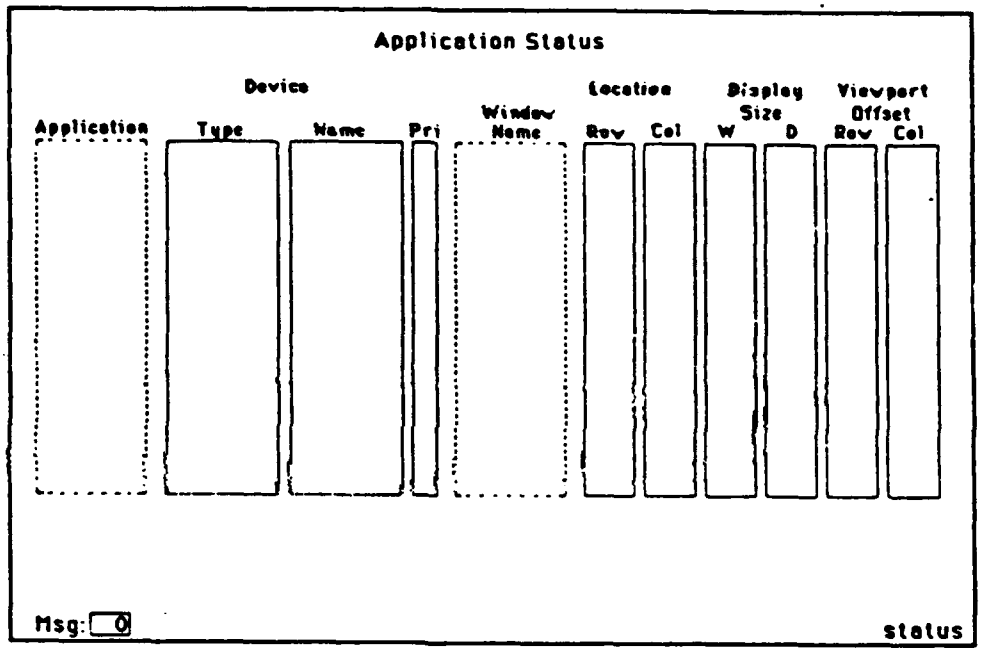


Figure 3-3 Application Status Form

The following is an explanation of the fields on this form.

Application — The name of the application using the window. This field cannot be changed.

Device Type — The device driver type which the application is running. This device driver is a special application which allows the application to communicate with the terminal. It must be an application name recognized by the NTM being used.

Device Name	The name of the port for the physical device that the application is running on. It must agree with the device type named. This value may be changed to move the application to any other device that is hardwired to the system. The device type field value must be appropriate for the device name.
Priority	The number that represents the order in which the windows are stacked on the screen. The last application to be initiated has a priority number of 1. It is on top of the stack of initial windows and is totally visible. The priority can be changed to give any application top priority. This has the same effect as selecting a winndow using the <SELECT> key in Window Manager mode.
Window Name	The name of the window or windows in the current stack that can be manipulated for each application. These names cannot be changed.
Location	The physical location on the screen of the upper Row/Col left corner of the window relative to the upper left corner of the containing window. The row and column values can be changed to change the location of the windows on the screen.
Display Size Width/Depth	The physical size of the window on the screen expressed in terms of columns wide and rows deep. Window sizes can be changed. This includes restoring visibility by giving dimension to a window that has been hidden (i.e., its size is 0 0).
Viewport Offset Row/Col	The row and column of the form displayed in row 1, column 1 of its containing window. If the offset is 0 0, then the upper left corner of the form is in the upper left corner of the window. Offsets can be changed to scroll the form currently displayed in the window.

The <ABORT> key aborts an application. Pressing this function key causes the NTM to be notified via a SIGABT call that the application containing the cursor should be terminated. The application must be defined in the NTM tables as being able to be terminated in this manner. For applications that been removed from the user's terminal screen or directed to another device, this key can also be used from the Application Status Form. When the <ABORT> key is pressed, the application named in the line containing the cursor is aborted.

3.2.6 Scripting

The Form Processor has the capability of recording all user input from a session in a script file. The script file can then be played back to duplicate the session. The Form Processor also has the capability of recording all output from a session in a save file. The combination of these capabilities allow for regression testing. By creating a script file and a save file

and later playing back the script file and generating another save file, the save files can be compared to detect any differences in the behavior of an application or the Form Processor itself. Scripting functions are specified in the startup message from the device driver.

3.3 Special Requirements

3.3.1 Programming Methods

The Form Processor is programmed using structured design and coding techniques. Basic programming standards for readability and ease of debugging are followed. The FP is implemented using the C programming language to insure portability of the FP code with minimum effort.

3.3.2 Expandibility

The modular design of the Form Processor allows new functionality to be added simply. Additional functions may be new callable routines, additional types of NTM messages, or additional function keys (including new modes).

3.4 Human Performance

Performance requirements for the FP have not yet been determined. These requirements should include a statement of response time for displaying and entering a screen and the ease of use of the features such as function keys.

3.5 Data Base Requirements

3.5.1 Sources and Types of Input

3.5.1.1 Form Definition File

The definition of each form is contained in a form definition file. This file contains a record specifying the file structure and the linearized version of the internal representation of the form.

3.5.1.2 Message Definition File

The definitions of routine status codes are contained in message definition files. Each file contains the code number, code name, and descriptive text for a block of 100 messages. The file name indicates the contained codes (e.g., MSG703 contains codes 70300 through 70399). These files are used to display messages to the user. The MSGLIN structure specifies the message file record format.

3.5.1.3 User Interface Database

The User Interface database contains the authorization and descriptive information required by the Form Processor. This includes:

- o a definition of each authorized user
- o the roles valid for each user
- o the functions valid for each role
- o a definition of each available function

The UI Database comprises four files: the user definition file, USRDEF, the user-role file, USEROL, the role-application file, ROLAPP, and the application definition file, APPDEF. Since the C programming language does not support indexed files, access to these files is provided by COBOL routines.

User Definition File

The user definition file contains information about each user of the IISS system. This is an indexed file named USRDEF which has one key which must be unique.

Application Definition File

The application definition contains information about each function which is available on IISS. It is an indexed file named APPDEF containing one key which must be unique.

User - Role File

The user - role file specifies the roles that are valid for each user of the IISS system. This is an indexed file named USEROL which contains two keys; although both keys can contain duplicates, the entire record must be unique.

Role - Application File

The role - application file specifies the applications that are valid for each role. This is an indexed file named ROLAPP which contains two keys; although both keys can contain duplicates, the entire record must be unique.

3.5.2 Destinations and Types of Output

The Form Processor generates the messages described in Section 3.2.2 and FDL and FD files when forms are created and modified at execution time. The structure of FDL files is described in the Forms Language Compiler Development Specification. The FD file structure is contained in Section 3.5.1.1.

3.5.3 Internal Tables and Parameters

All of the Form Processor's internal data is collected into a network of statically and dynamically allocated structures which are linked by pointers. Access to the network is through the root structure called UID.

The MSGFIL structure contained in UID relates error numbers to message files.

Information about each logged on user is kept in a USR structure. UID points to a list of these structures. The USR structure points to lists of the logical devices, applications, physical devices, and messages belonging to the user.

Information about each physical device being used is kept in a PD structure. The UID points to a list of all of these structures, and USR points to those which belong to the same user. The PD structure points to a list of the logical devices associated with the physical device.

Information about each running application is kept in the AP structure. The UID points to a list of all of these structures, and USR points to those which belong to the same user. The AP structure points to the logical devices associated with it.

Information about logical devices is represented by the FPD structure. The USR structure points to the UI logical device (i.e., the logical device where system forms such as the IISS logon and function screens are displayed). The PD structure points to the logical devices associated with the physical device and the AP structure points to the logical devices associated with it.

Forms are represented by a tree of FIELD structures. The FPD structure points to the forms which are currently open and the current display. The FIELD structure points to its prompt fields and other fields which are contained in it (e.g., a form field points to the fields on the form and a window field points to the forms in the window). The FIELD structure contains either a FORM, WINDOW, ITEM, ARRAY, or PROMPT structure depending on the type of field.

The RELPOS structure contained in FIELD is used to represent relative positions. It points to the field being referenced and each FIELD structure points to a list of RELPOS structures which reference it.

The FORM structure points to all the displayed instances of the form and a list of attributes for the form.

The ITEM structure points to the help message for the item and a data buffer which contains the current and previous values of the item.

The PROMPT structure points to the prompt text.

The ATTMAP structure is used to represent background and display attributes. FORM structures point to the attributes which are defined for the form and FIELD structures point to the background or display attribute defined for the field.

The MSGBUF structure is used to represent messages. USR points to all the messages for the user, and FPD points to the messages for the logical device.

The ENODE structure is used to represent field values. The FIELD structure points to a tree of ENODE structures which represent the expression specified as the value of the field. ENODE structures point to other ENODE structures and to fields being referenced. A FIELD structure points to a list of all ENODE structures which reference it.

The union EXPARG contained in the ENODE structure represents expression arguments.

The union EXPVAL is used by reutines which evaluate expressions to represent the expression value.

The POSITION structure contained in many of the other structures represents a position as a specific row and column.

The SIZE structure contained in many of the other structures represents size as a number of columns wide and rows deep.

3.5.4 PHIGS Data Structures

The following sections list the contents of the PHIGS data structures. Each structure includes the following information:

- o The name of the element.
- o The coordinate system (if appropriate)
- o The allowable values.
- o The data type (integer, character, etc.)
- o The initial value if needed.

The following is a description of the notation used to in the structure list:

I	integer	whole number
R	Real	floating point number
S	string	sequence of ASCII characters.
P2	2D point	2 real values specifing the x= and y -coordinates of a location.
P3	3D point	three dimensional location (as above)
N	Name	identification of a workstation or a number of primitives
E	Enumeration type	an identifier which is a enumeration of the value represented.
NS	name set	A data type consisting of a set of entries with one entry corresponding to each item of an enumeration type.
PP	pick	A compound data type containing the following items: structure identifier (I), pick identifier (N), and element number (I).
ER	executive reference	A compound data type containing the following items: a point (P3), a normal vector (3xR).
C	connector	identifier
F	file	
W	workstation type	

For reference purposes, the following coordinate systems abbreviations are provided:

MC Modelling coordinating system
WC World coordinate system
VRC Viewing reference coordinate system
NPC Normalized projection coordinate system
DC Device coordinate system

An occurence of n (as in 1..n) indicates the variable integer value.

Initial values which may be required occur on the last column of the data structure list. The following are the appropriate abbreviations:

undef undefined value.
i.d. implementation dependent
p.d.t. initial value taken from the PHIGS description table
w.d.t. initial value taken from the workstation description table.
empty list contains no values.

3.5.4.1. PHIGS Description Table

The Description Table contains the supported miniumu functions (as specified in section 3.1.1.) and the allowable limits for each function. The following is a list of the table elements:

number of available workstation types	(1..n)	I	i.d.
list of available workstationtypes	L(W)		i.d.
maximum number of simultaneously open workstations	(1..n)	I	i.d.
maximum number of simultaneously open archive files	(1..n)	I	i.d.
number of available character sets	(1..n)	I	i.d.
list of available character sets (The first entry designates the ANSI X3.4-1977 character set, represented as character set 0. The coorespondence between character set identifer 1 through n and their respective character set definitions is defined in implementation documentation.)	(1..n)	I	i.d.
default polyline index	(1..n)	I	1
default linetype	(-n..n)	I	1
default linewidth scale factor		R	1.0
default polyline colour index	(0..n)	I	1
default linetype ASF	(BUNDLED, INDIVIDUAL)	E	see note 1
default linewidth scale factor ASF	(BUNDLED, INDIVIDUAL)	E	see note 1
default polyline colour index ASF	(BUNDLED, INDIVIDUAL)	E	see note 1
default polymarker index	(1..n)	I	1
default marker type	(-n..n)	I	3
default marker size scale factor		R	1.0
default polymarker colour index	(0..n)	I	1
default marker type ASF	(BUNDLED, INDIVIDUAL)	E	see note 1
default marker size scale factor ASF			

	(BUNDLED, INDIVIDUAL)		E	see note 1
default polymarker colour index ASF				
	(BUNDLED, INDIVIDUAL)		E	see note 1
default text index		(1..n)	I	1
default text font		(1..n)	I	1
default text precision	(STRING,CHAR,STROKE)		E	STRING
default character expansion factor			R	1.0
default character spacing			R	0.0
default text colour index		(0..n)	I	1
default text font ASF	(BUNDLED, INDIVIDUAL)		E	see note 1
default text precision ASF	(BUNDLED, INDIVIDUAL)		E	see note 1
default character expansion factor ASF				
	(BUNDLED, INDIVIDUAL)		E	see note 1
default character spacing ASF	(BUNDLED, INDIVIDUAL)		E	see note 1
default text colour index ASF	(BUNDLED, INDIVIDUAL)		E	see note 1
default character height			R	0.01
default character up vector			2xR	(0.0,1.0)
default text path	(RIGHT,LEFT,UP,DOWN)		E	RIGHT
default text alignment (horizontal & vertical)				
(NORMAL,LEFT,CENTRE,RIGHT;NORMAL,TOP,CAP,HALF,BASE, BOTTOM)			2xE	
default annotationtext character height			R	0.01
default annotation text character up vector			2xR	(0.0,1.0)
default annotation text character width			R	0.01
default annotation text character base vector			2xR	(1.0,0.0)
default annotation text path	(RIGHT,LEFT,UP,DOWN)		E	RIGHT
default annotation text alignment (horizontal & vertical)				
(NORMAL,LEFT,CENTRE,RIGHT;NORMAL,TOP, CAP,HALF,BASE,BOTTOM)			2xE	(NORMAL,NORMAL)
default annotation style		(-n,n)	I	1
default interior index		(1..n)	I	1
default interior style (HOLLOW,SOLID,PATTERN, HATCH,EMPTY)			E	HOLLOW
default interior style index		(-n..n)	I	1
default interior colour index		(0..n)	I	1
default interior style ASF (BUNDLED,INDIVIDUAL)			E	see note 1
default interior style index ASF (BUNDLED, INDIVIDUAL)			E	see note 1
default interior colour index ASF (BUNDLED,INDIVIDUAL)			E	see note 1
default edge flag		(OFF,ON)	E	OFF
default edgetype		(-n..)	I	1
default edgewidth scale factor			R	1.0
default edge colour index		(0..n)	I	1
default edge flag ASF (BUNDLED, INDIVIDUAL)			E	see note 1
default edgetype ASF (BUNDLED, INDIVIDUAL)			E	see note 1
default edgewidth scale factor ASF (BUNDLED,INDIVIDUAL)			E	see note 1
default edge colour index ASF (BUNDLED,INDIVIDUAL)			E	see note 1
default pattern size	MC SX, SY>0		2xR	(1.0,1.0)
default pattern reference point	MC		P3	(0.0,0.0,0.0)
default pattern reference vectors	MC		2xP3	(1.0,0.0,0.0)
	MC			(0.0,1.0,0.0)
default pick identifier			N	see note 2
default view index		(0..n)	I	0
default HLHSR identifier		(0..n)	I	0
default name set			NS	no classes
default global modelling transformation			4x4xR	Identify
default local modelling transformation			4x4xR	Identify
default modelling clipping volume	MC			all of WC space

default modelling clipping indicator (CLIP,NOCLIP)	E	NOCLIP
number of available generalized structure elements (0..n)	I	i.d.
list of available generalized structure elements (may be empty)		
for every GSE:	N	i.d.
workstation dependency indicator (WORKSTATION-INDEPENDENT,WORKSTATION-DEPENDENT)	E	i.d.
number of available modelling clipping half-spaces (6..n)	I	i.d.
number of available modelling operators (4..16)	I	i.d.
list of available modelling clipping operators (0..15)	nxI	(1,3,5,15)

NOTE 1 - All of the initial ASF values are the same. They are all INDIVIDUAL

NOTE 2 - The pick identifier default is the language dependent equivalent of zero (0).

3.5.4.2 PHIGS Traversal State List

The traversal state list is created each time a traversal is initiated. the initial values for the state list are copied from the PHIGS description table. The PHIGS traversal state list exist only during a traversal. The values in the PHIGS traversal state list cannot be read.

current polyline index (1..n)	I	p.d.t.
current linetype (-n..n)	I	p.d.t.
current linewidth scale factor	R	p.d.t.
current polyline colour index (0..n)	I	p.d.t.
current linetype ASF (BUNDLED,INDIVIDUAL)	E	p.d.t.
current linewidth scale factor ASF(BUNDLED,INDIVIDUAL)	E	p.d.t.
current polyline colour index ASF (BUNDLED,INDIVIDUAL)	E	p.d.t.
current polymarker index (1..n)	I	p.d.t.
current marker type (-n..n)	I	p.d.t.
current marker size scale factor	R	p.d.t.
current polymarker colour index (0..n)	I	p.d.t.
current marker types ASF (BUNDLED,INDIVIDUAL)	E	p.d.t.
current marker size scale factor ASF (BUNDLED,INDIVIDUAL)	E	p.d.t.
current polymarker colour index ASF(BUNDLED,INDIVIDUAL)	E	p.d.t.
current text index (1..n)	I	p.d.t.
current text font (1..n)	I	p.d.t.
current text precision (STRING,CHAR,STROKE)	E	p.d.t.
current character expansion factor	R	p.d.t.
current character spacing	R	p.d.t.
current text colour index (0..n)	I	p.d.t.
current text font ASF (BUNDLED,INDIVIDUAL)	E	p.d.t.
current text precision ASF (BUNDLED,INDIVIDUAL)	E	p.d.t.
current character expansion factor ASF (BUNDLED,INDIVIDUAL)	E	p.d.t.
current character spacing ASF (BUNDLED,INDIVIDUAL)	E	p.d.t.
current text colour index ASF (BUNDLED,INDIVIDUAL)	E	p.d.t.
current character height	R	p.d.t.
current character up vector	2xR	p.d.t.
current character width	R	p.d.t.
current character base vector	2xR	p.d.t.
current text path (RIGHT,LEFT,UP,DOWN)	E	p.d.t.

current text alignment (horizontal & vertical) (NORMAL,LEFT,CENTRE,RIGHT;NORMAL,TOP,CAP,HALF, BASE,BOTTOM)		2xE	p.d.t.
current annotation text character height		R	p.d.t.
current annotation text character up vector		2xR	p.d.t.
current annotation text width		R	p.d.t.
current annotation text path	(RIGHT,LEFT,UP,DOWN)	E	p.d.t.
current annotation allignment (horizontal & vertical) (NORMAL,LEFT,CENTRE,RIGHT;NORMAL,TOP,CAP,HALF, BASE,BOTTOM)		2xE	p.d.t.
current annotation style	(-n,n)	I	p.d.t
current interior index	(1..n)	I	p.d.t.
current interior style (HOLLOW,SOLID,PATTERN, HATCH,EMPTY)		E	p.d.t.
current interior style index	(-n..n)	I	p.d.t.
current interior colour index	(0..n)	I	p.d.t.
current interior style ASF	(BUNDLED, INDIVIDUAL)	E	p.d.t.
current interior style index ASF	(BUNDLED, INDIVIDUAL)	E	p.d.t.
current interior colour index ASF	(BUNDLED, INDIVIDUAL)	E	p.d.t.
current edge flag	(OFF,ON)	E	p.d.t.
current edgetype	(-n..n)	I	p.d.t.
current edgewidth scale factor		R	p.d.t.
current edge colour index	(0..n)	I	p.d.t.
current edge flag ASF	(BUNDLED, INDIVIDUAL)	E	p.d.t.
current edgetype ASF	(BUNDLED, INDIVIDUAL)	E	p.d.t.
current edgewidth scale factor ASF	(BUNDLED, INDIVIDUAL)	E	p.d.t.
current edge colour index ASF	(BUNDLED, INDIVIDUAL)	E	p.d.t.
current pattern size	MC SX,SY>0	2xR	p.d.t.
current pattern reference point	MC	P3	p.d.t.
current pattern reference vectors	MC	2XP3	p.d.t.
current pick identifier		N	p.d.t.
current view index	(0..n)	I	p.d.t.
current HLHSR identifier	(0..n)	I	p.d.t.
current name set		NS	p.d.t.
current global modelling transformation		4x4xR	p.d.t.
current local modelling transformation		4x4xR	p.d.t.
current modelling clipping volume	MC L(O,L(HS))		p.d.t.
curent modelling clipping indicator	(CLIP,NOCLIP)	E	p.d.t.

3.5.4.3 PHIGS State List

The following is a description of the PHIGS state list.

set of open workstations		nxN	empty
name of open structure		I	empty
current element pointer	(0..n)	I	0
edit mode	(INSERT,REPLACE)	E	INSERT
list of structure identifiers		L(I)	empty
archive file list (one entry for each open archive file)			emtpy
archive file		L(F)	undef
archice file identifier		L(N)	undef
archival conflict resolution flag	(MAINTAIN, ABANDON, UPDATE)	E	UPDATE
retrieval conflict resolution flag	(MAINTAIN, ABANDON,		

	UPDATE)	E	ABANDON
input queue (one entry for each event report)			empty
for every entry:			
workstation identifier		N	undef
device number	(1..n)	I	undef
last of group of simultaneous events	(NOTLAST,LAST)	E	undef
(a single event is indivated by last)			
input class (LOCATOR,STROKE,VALUATOR,CHOICE,			
PICK,STRING)		E	undef
if LOCATOR			
view index	(0..n)	I	undef
position	WC	P3	undef
if STROKE			
view index	(0..n)	I	undef
list of points in stroke	WC	L(P3)	undef
if VALUATOR			
value		R	undef
if CHOICE			
status	(OK,NOCHOICE)	E	undef
choice number	(0..n)	I	undef
if PICK			
status	(OK,NOPICK)	E	undef
pick path		L(PP)	undef
if STRING			
string		S	undef
current event report containing:			
input class (NONE,LOCATOR,STROKE,VALUATOR,CHOICE,			
PICK,STRING)		E	none
if LOCATOR			
view index	(0..n)	I	undef
position	WC	P3	undef
if STROKE			
view index	(0..n)	I	undef
list of points in stroke	WC	L(P3)	undef
if VALUATOR			
value		R	undef
if CHOICE			
status	(OK,NOCHOICE)	E	undef
choice number	(0..n)	I	undef
if PICK			
status	(OK,NOPICK)	E	undef
pick path		L(PP)	undef
if STRING			
string		S	undef
more simultaneous events	(NOMORE,MORE)	E	NOMORE

3.5.4.4 PHIGS Workstation State List

One workstation state list exists for every open workstation.

workstation identifier	N
connection identifier	C
workstation type	W

The above 3 entries are initialized by OPEN WORKSTATION.

Entries in this group do not exist for workstations of category MI.

The initial value of view table entries may be predefined. The initial value of view table entries that are not predefined is the same as view table entry 0.

number of view table entries	(6..n)	I	w.d.t.
table of defined view representations ordered by view transformation input priority (initially in numerical order with view index 0 highest priority) for every entry:			
view index	(0..n)	I	w.d.t.
view transformation update state	(NOTPENDING,PENDING)	E	NOTPENDING
current view orientation matrix	4x4xR		w.d.t.
current view mapping matrix	4x4xR		w.d.t.
current view clipping limits	6xR		w.d.t.
current x-y clipping indicator	(CLIP,NOCLIP)	E	w.d.t.
current back clipping indicator	(CLIP,NOCLIP)	E	w.d.t.
current front clipping indicator	(CLIP,NOCLIP)	E	w.d.t.
requested view orientation matrix	4x4xR		w.d.t.
requested view mapping matrix	4x4xR		w.d.t.
requested view clipping limits	6xR		w.d.t.
requested x-y clipping indicator	(CLIP,NOCLIP)	E	w.d.t.
requested back clipping indicator	(CLIP,NOCLIP)	E	w.d.t.
requested front clipping indicator	(CLIP,NOCLIP)	E	w.d.t.
HLHSR update state	(NOTPENDING,PENDING)	E	NOTPENDING
current HLHSR mode	(-n..n)	I	0
requested HLHSR mode	(-n..n)	I	0
transformation update state	(NOTPENDING,PENDING)	E	NOTPENDING
requested workstation window	NPC 6xR		(0,1,0,1,-1,0)
current workstation window	NPC 6xR		(0,1,0,1,-1,0)
requested workstation viewport	DC 6xR		max display space from w.d.t.
current workstation viewport	DC 6xR		max display space from w.d.t.
Entries in this group do not exists for workstations of category INPUT and MI			
table of posted structures			empty
For every entry:			
structure ID		I	
posted priority		R	
deferral mode	(ASAP,BNIG,BNIL,ASTI,WAIT)	E	w.d.t.
modification mode	(NIVE,UWOR,UQUM)	E	w.d.t.
display surface empty	(EMPTY,NOTEMPTY)	E	EMPTY
state of visual representation	(CORRECT, DEFERRED,SIMULATED)	E	CORRECT
number of polyline bundle table entries	(5..n)	I	w.d.t.
table of defined polyline bundes for every entry:			
polyline index	(1..n)	I	w.d.t.
linetype	(-n..n)	I	w.d.t.
linewidth scale factor		R	w.d.t.
polyline colour index	(0..n)	I	w.d.t.
number of polymarkder bundle table entires	(5..n)	I	w.d.t.
table of defined polymarker bundles			

for every entry:				
polymarker index		(1..n)	I	w.d.t.
marker type		(-n..n)	I	w.d.t.
marker size scale factor			R	w.d.t.
polymarker colour index		(0..n)	I	w.d.t.
number of text bundle table entries		(5..n)	I	w.d.t.
table of defined text bundles				
for every entry:				
text index		(1..n)	I	w.d.t.
text font		(-n..n)	I	w.d.t.
text precision	(STRING,CHAR,STROKE)		E	w.d.t.
character expansion factor			R	w.d.t.
character spacing			R	w.d.t.
text colour index		(0..n)	I	w.d.t.
number of interior bundle table entries		(5..n)	I	w.d.t.
table of defined interior bundles				
for every entry:				
interior index		(1..n)	I	w.d.t.
interior style	(HOLLOW,SOLID,PATTERN,HATCH,EMPTY)			E w.d.t.
interior style index		(-n..n)	I	w.d.t.
interior colour index		(0..n)	I	w.d.t.
number of edge bundle table entries		(5..n)	I	w.d.t.
table of defined edge bundles				
for every entry:				
edge index		(1..n)	I	w.d.t.
edge flag		(OFF,ON)		E w.d.t.
edgetype		(-n..n)	I	w.d.t.
edgewidth scale factor			R	w.d.t.
edge colour index		(0..n)	I	w.d.t.
number of pattern table entries		(0,1..n)	I	w.d.t.
table of pattern representations				
for every entry:				
pattern index		(1..n)	I	w.d.t.
pattern array dimensions		(1..n)	2xI	w.d.t.
pattern array		(0..n)	nxnxI	w.d.t.
current colour model		(-n..n)	I	w.d.t.
number of colour table entries		(2..n)	I	w.d.t.
table of colour representations				
for every entry:				
colour index		(0..n)	I	w.d.t.
colour (component triplet)		[0,1]	3xR	w.d.t.
highlighing inclusion set			NS	no classes
highlighting exclusion set			NS	no classes
invisibility inclusion set			NS	no classes
invisibility exclusion set			NS	no classes

Entries in this group do not exist for workstation of category OUTPUT, MO and MI

for every logical input device of class LOCATOR				
locator device number		(1..n)	I	w.d.t.
operating mode	(REQUEST,SAMPLE,EVENT)		E	REQUEST
echo switch	(ECHO,NOECHO)		E	ECHO
initial view index		(0..n)	I	0
initial locator position		WC	P	w.d.t.
prompt and echo type		(-n..n)	I	1
echo volume		DC	6xR	w.d.t.
locator data record			D	i.d.

```
for every logical input device of class STROKE:
     stroke device number                              (1..n)  I    w.d.t.
     operating mode                 (REQUEST,SAMPLE,EVENT)      E
REQUEST
     echo switch                              (ECHO,NOECHO)      E   ECHO
     initial value                                        R     w.d.t.
     prompt and echo type                          (-n..n) I    1
     echo volume                                   DC     6xR   w.d.t.
     valuator data record containing at least             D     i.d.
       low value                                          R     w.d.t.
       high value                                         R     w.d.t.

for every logical input device of class CHOICE:
     choice device number                              (1..n)  I    w.d.t.
     operating mode                 (REQUEST,SAMPLE,EVENT)      E
REQUEST
     echo switch                              (ECHO,NOECHO)      E   ECHO
     initial status                            (OK,NOCHOICE) E NOCHOICE
     initial choice number                         (1..n)  I    undef.
     prompt and echo type                          (-n..n) I    1
     echo volume                                   DC     6xR   w.d.t.
     choice data record                                   D     i.d.

for every logical input deavice of class PICK:
     pick device number                                (1..n)  I    w.d.t.
     operating mode                 (REQUEST,SAMPLE,EVENT)      E
REQUEST
     echo switch                              (ECHO,NOECHO)      E   ECHO
     initial status                              (OK,NOPICK)     E
NOPICK
     pick inclusion set                                   NS all classes
                                                             detectable
     pick exclusion set                                   NS no classes
     initial pick path                                 L(pp)     empty
     prompt and echo type                          (-n..n) I    1
     echo volume                                   DC     6xR   w.d.t.
     pick data record                                     D     i.d.
   intial pick path order       (TOPFIRST,BOTTOMFIRST)    E    TOPFIRST

for every logical input device of class STRING:
     string device number                              (1..n)  I    w.d.t.
     operating mode                 (REQUEST,SAMPLE,EVENT)      E
REQUEST
     echo switch                              (ECHO,NOECHO)      E   ECHO
     initial string                                       S     empty
     prompt and echo type                          (-n..n) I    1
     echo volume                                   DC     6xR   w.d.t.
     string data record containing at least:              D     i.d.
       input buffer size (characters)              (1..n)  I    w.d.t.
       initial cursor position                     (1..n)  I    w.d.t.
```

3.5.4.5 PHIGS Worstation Description Table

The five types of workstations are:

- o OUTPUT (output only)
- o INPUT (input only)

- o OUTIN (both input and output)
- o MO (metafile output)
- o MI (metafile input)

The following descriptions may contain entries which are used in one or more of the above workstation types. More than one workstation identifier may belong to a workstation type, and a number of different workstation types may exist for each of the categories depending on the capabilities of particular physical device.

Entries in this group exist for all workstation categories

workstation type			W	i.d.
workstation category	(OUTPUT,INPUT,OUTIN,MO,MI)		E	

Entries in this group do not exist for workstations of category MO and MI

device coordinate units	(METRES,OTHER)		E	i.d.
maximum display space size (visible volume of the display surface or available volume for use of physical input devices for workstations of category INPUT)				
in device coordinates	DC	>0	3xR	i.d.
in device address units (integer by integer)		>0	3xI	i.d.
(for vector displays, for example, the device address units give the highest possible resolution; for raster displays, the number of columns and lines of the raster array)				
number of available HLHSR identifiers		(1..n)	I	i.d.
list of available HLHSR identifiers		(1n..n)	nxI	i.d.
number of available HLHSR modes		(1..n)	I	i.d.
list of available HLHSR modes		(-n..n)	nxI	i.d.
number of predefined view indices (representations)		(6..n)	I	i.d.
table of predefined view representations, for every entry:				
view orientation matrix			4x4xR	i.d.
view mapping matrix			6xR	i.d.
view clipping limits			E	i.d.
x-y clipping indicator	(CLIP,NOCLIP)		E	i.d.
back clipping indicator	(CLIP,NOCLIP)		E	i.d.
front clipping indicator	(CLIP,NOCLIP)		E	i.d.

Entries in this group do not exit for workstations of category INPUT

raster or vector display	(VECTOR,RASTER,OTHER)	E	i.d.
(VECTOR = vector device, RASTER = raster device, OTHER = other device, e.g., vector + raster)			
dynamic modification accepted for:			
view representation	(IRG,IMM,CBS)	E	i.d.
polyline bundle representation	(IRG,IMM,CBS)	E	i.d.
polymarker bundle representation	(IRG,IMM,CBS)	E	i.d.
text bundle representation	(IRG,IMM,CBS)	E	i.d.
interior bundle representation	(IRG,IMM,CBS)	E	i.d.
edge bundle representation	(IRG,IMM,CBS)	E	i.d.
pattern representatoin	(IRG,IMM,CBS)	E	i.d.
colour representation	(IRG,IMM,CBS)	E	i.d.

workstation transformation	(IRG,IMM,CBS)	E	i.d.
highlighting filter	(IRG,IMM,CBS)	E	i.d.
invisibility filter	(IRG,IMM,CBS)	E	i.d.
HLHSR mode	(IRG,IMM,CBS)	E	i.d.
(IRG:implicit regeneration necessary; IMM:performed immediately; CBS: can be simulated)			
default value for deferral state:			
deferral mode	(ASAP,BNIG,BNIL,ASTI,WAIT)	E	i.d.
modification mode	(NIVE,UWOR,UQUM)	E	i.d.
number of available linetypes	(-n..-4,4..n)	I	i.d.
(a negative value returned indicates that the characteristics of the implementation dependent linetypes are derived from the linetype values directly. The absolute balue of the value returned indicates the number of registered linetypes supported.)			
list of available linetypes	(-n..n)	L(I)	i.d.
number of available linewidths	(0..n)	I	i.d.
(a value of 0 indicates that a continuous range of linewidths is supported)			
nominal linewidth	DC >0	R	i.d.
minimum linewidth	DC >0	R	i.d.
maximum linewidth	DC >0	R	i.d.
number of predefined polyline indices (bundles)	(5..n)	I	i.d.
table of predefined polyline bundles			
for every entry:			
linetype	(-n..n)	I	i.d.
linewidth scale factor		R	i.d.
polyline colour index (within range of predefined colour indices)	(0..n)	I	i.d.
number of available marker types	(-n..-4,4..n)	I	i.d.
(a negative value returned indicates that the characteristics of the implementation dependent marker types are derived from the marker type values directly. The absolute value of the value returned indicates the number of registered marker types supported)			
list of available marker types	(-n..n)	L(I)	i.d.
number of available marker sizes	(0..n)	I	i.d.
(a value of 0 indicates that a continuous range of marker sizes is supported)			
nominal marker size	DC >0	R	i.d.
minimum marker size	DC >0	R	i.d.
maximum marker size	DC >0	R	i.d.
number of predefined polymarker indices(bundles)	(5..n)	I	i.d.
table of predefined polymarker bundles.			
for each entry:			
marker type	(-n..n)	I	i.d.
marker size scale factor		R	i.d.
polymarker colour index(within range of predefined colour indices)	(0..n)	I	i.d.
number of text font and precision pairs	(4..n)	I	i.d.
list of text font and precision pairs	(-n..n;STRING, CHAR,STROKE)	L(I,E)	i.d.
number of available character expansion factors	(0..n)	I	i.d.
(a value of 0 indicates that a continuous range of character expansion factors are supported)			
minimum character expansion factor	>0	R	i.d.

maximum character expansion factor		>0	R	i.d.	
number of available character heights		(0..n)	I	i.d.	
(a value of 0 indicates that a continous range of character heights is supported.)					
minimum character height	DC	>0	R	i.d.	
maximum character height	DC	>0	R	i.d.	
number of predefined text indices(bundles)		(6..n)	I	id.d	
table of predefined text bundles					
for every entry:					
text font		(-n..n)	I	i.d.	
text precision	(STRING,CHAR,STROKE)		E	i.d.	
character expansion factor			R	i.d.	
character spacing			R	i.d.	
text colour index (within range of predefined colour indices)		(0..n)	I	i.d.	
number of available annotation styles		(2..n)	I	i.d.	
list of availalbe annotation styles		(-n..n)	L(I)	i.d.	
number of available interior styles		(1..5)	I	i.d.	
list of available interior styles (HOLLOW,SOLID, PATTERN,HATCH,EMPTY)			L(E)	i.d.	
number of available hatch styles	(-n..-3,0,3..n)		I	i.d.	
(a negative value returned indicates that the characteristics of the implementation dependent hatch styles are derived from the hatch style values directly. The absolute value of the value returned indicates the number of registered hatch styles supported.)					
list of available hatch styles		(-n..n)	L(I)	i.d.	
number of predefined interior indices(bundles)		(5..n)	I	i.d.	
table of predefined interior bundles					
for every entry:					
interior style (HOLLOW,SOLID,PATTERN,HATCH, EMPTY)			E	i.d.	
interior style index		(-n..n)	I	i.d.	
(for interior style PATTERN is within the range of predefined pattern indices)					
(for interior style HATCH is within the range of predefined pattern indices)					
interior colour index (within the range of predefined colour indices		(0..n)	I	i.d.	
number of available edgetypes	(-n..-1,1..n)		I	i.d.	
(a negative value returned indicates that the characteristics of the implementation dependent edgetypes are derived from the edgetype values directly. The absolute value of the value returned indicates the number of registered edgetypes supported.)					
list of available edgetypes		(-n..n)	L(I)	i.d.	
number of available edgewidths		(0..n)	I	i.d.	
(a value of 0 indicates that a continuous range of edgewidths is supported)					
nominal edgewidth	DC	>0	R	i.d.	
minimum edgewidth	DC	>0	R	i.d.	
maximum edgewidth	DC	>0	R	i.d.	
number of predefined edge indices(bundles)		(5..n)	I	i.d.	
table of predefined edge bundles					
for every entry:					
edge flag		(OFF,ON)	E	i.d.	
edgetype		(-n..n)	I	i.d.	
edgewidth scale factor			R	i.d.	
edge colour index (within range of predefined					

colour indices)	(0..n)	I	i.d.
number of predefined pattern indices (representations)	0..n)	I	i.d.
table of predefined pattern representations for every entry:			
pattern array dimensions	(1..n)	2xI	i.d.
pattern array	(o..n)	nxnxI	i.d.
number of available colour models	(2..n)	I	i.d.
list of available colour models	(-n..n)	L(I)	i.d.
default colour model	(-n..n)	I	i.d.
number of available colours or intensities (a value of 0 indicates that a continuous range of colours is supported)	(0,2..n)	I	i.d.
colour available (COLOUR,MONOCHROME)		E	i.d.
number of predefined colour indices (representations)	(2..n)	I	i.d.
table of predefined colour representations for every entry:			
colour (component triplet)	[0,1]	3xR	i.d.
number of available generalized drawing primitives 3 (GDP3) (may be empty) for every EDP3:			
GDP3 identifier		N	i.d.
number of sets of attributes used	(0..5)	I	i.d.
list of sets of attributes used (POLYLINE,POLYMARKER,TEXT,INTERIOR,EDGE)		L(E)	i.d.
number of available generalized drawing primitives (GDP)	(0..n)	I	i.d.
list of available generalized drawing primitives (GDP) (may be empty) for every GDP:			
GDP identifier		N	i.d.
number of sets of attributes used	(0..5)	I	i.d.
list of sets of attributes used (POLYLINE,POLYMARKER,TEXT,INTERIOR,EDGE)		L(E)	i.d.
number of available generalized structure elements	(0..n)	I	i.d.
list of available generalized structure elements (may be empty) for every GSE:			
GSE identifier		N	i.d.
number of structure priorities supported (a value of 0 indicates that a continuous range of priorities is supported)	(0,2..n)	I	i.d.
maximum number of polyline bundle table entries	(20..n)	I	i.d
maximum number of polymarker bundle table entries	(20..n)	I	i.d.
maximum number of text bundle table entries	(20..n)	I	i.d.
maximum number of interior bundle table entries	(20..n)	I	i.d.
maximum number of edge bundle table entries	(20..n)	I	i.d.
maximum number of hatch styles	(3..n)	I	i.d.
maximum number of pattern table entries	(0,10..n)	I	i.d.
maximum number of colour indices	(2..n)	I	i.d.
maximum number of view indices	(6..n)	I	i.d.
dynamic modification accepted for:			
structure content modification	(IRG,IMM,CBS)	E	i.d.
post structure	(IRG,IMM,CBS)	E	i.d.

unpost structure	(IRG,IMM,CBS)	E	i.d.
delete structure	(IRG,IMM,CBS)	E	i.d.
reference modifications	(IRG,IMM,CBS)	E	i.d.

where:
IRG: Implicit regeneration necessary (may be deferred)
IMM: Performed immediately
CBS: Can be simulated

Entires in this group do not exist for workstations of category OUTPUT, MO or MI

number of logical input devices of class LOCATOR	(0..n)	I	i.d.
for every logical input device of class LOCATOR:			
locator device number	(1..n)	I	i.d.
default intial locator position	WC	P3	i.d.
number of available prompt and echo types	(1..n)	I	i.d.
list of available promt and echo types	(-n..n)	L(I)	i.d.
default echo volume	DC	6xR	i.d.
default locator data record		d	i.d.
number of logical input devices of class STROKE	(0..n)	I	i.d.
for every logical input device of class STROKE:			
stroke device number	(1..n)	I	i.d.
maximum input buffer size	(64..n)	I	i.d.
number of available prompt and echo types	(1..n)	I	i.d.
list of available prompt and echo types	(-n..n)	L(I)	i.d.
default echo volume	DC	6xR	i.d.
default stroke data record containing at least:		D	i.d.
input buffer size	(1..n)	I	i.d.
number of logical input devices of class VALUATOR	(0..N)	I	i.d.
for every logical device of class VALUATOR			
valuator device number	(1..n)	I	i.d.
default initial value		R	i.d.
number of available prompt and echo types	(1..n0	I	i.d.
default echo volume	DC	6xR	i.d.
default valuator data record containing at least		D	i.d.
low value		R	i.d.
high value		R	i.d.
number of logical input devices of class CHOICE	(0..n)	I	i.d.
for every logical input device of class CHOICE			
choice device number	(1..n)	I	i.d.
maximum number of choice alternatives	(1..n)	I	i.d.
number of available prompt and echo types	(1..n)	I	i.d.
list of available prompt and echo types	(-n..n)	L(I)	i.d.
default echo volume	DC	6xR	i.d.
default choice data record		D	i.d.
number of logical input devies of class PICK	(0..n)	I	i.d.
for every logical input device of class PICK:			
pick device number	(1..n)	I	i.d.
number of available prompt and echo types	(1..n)	I	i.d.
list of available prompt and echo types	(-n..n)	nxI	i.d.

```
     default echo volume                          DC          6xR    i.d.
     default pick data record containing at least:             D     i.d.
       pick path order    (TOPFIRST,BOTTOMFIRST)                    E
TOPFIRST

number of logical input devices of class STRING (0..n)       I      i.d.
for every logical input device of class STRING:
     string device number                        (1..n)      I      i.d.
     maximum input buffer size                   (72.n)      I      i.d.
     number of available prompt and echo types   (1..n)      I      i.d.
     list of available prompt and echo types     (-n..n)     L(I)   i.d.
     default echo volume                          DC          6xR    i.d.
default straing data record containing at least:             D     i.d.
  input buffer size (characters)                 (1..n)      I      i.d.
  initial cursor position                        (1..n)      I      i.d.
```

SECTION 4

QUALITY ASSURANCE PROVISIONS

4.1 Introduction and Definitions

The FP is tested using the following tests:

Computer programming test and evaluation. This testing primarily involves testing all the FP interface routines and internal functions for correct processing and output.

System test. This testing involves testing all the FP interface routines and internal functions within the integrated system.

4.2 Computer Programming Test and Evaluation

The test developed for the FP consists of another computer program which uses the FP interfaces routines by an application program. Every FP interface routine is exercised directly by the computer test program and every internal function is indirectly exercised by the computer test program. Variations on the computer test program may be provided by user interaction with the computer test program.

The same computer test program is run to test the FP during the system test process with the other components of the IISS.

SECTION 5

PREPARATION FOR DELIVERY

The implementation site for the constructed software is the ICAM Integrated Information Support System (IISS) Testbed site located at Arizona State University, Tempe, Airzona. The software associated with each FP CPCI release is delivered on a media which is compatible with the IISS Testbed. The release is clearly identified and includes instructions on procedures to be followed for installation of the release.

APPENDIX A

FP ROUTINE RETURN CODES

The following errors are warning errors:

Name	Message
OK	(Blank - normal completion)
FPMSGS	Form Processor Messages
INVPAG	Invalid page number
FNOTFND	Form not found
FISOPEN	Form is already open
ALCERR	Memory allocation error
OPNERR	Open error - Unable to read form definition file
EXPERR	Error in expression
TRNCFLD	Field value too long - truncated
UNKTYPE	Internal error - Unknown field type
PNOTARY	Qualified name is not an array
NILKEY	Non-functional key
SYSERR	System error - Call System Administrator
PBFULL	Paste buffer full
INVEDT	Field can not be edited
FILRERR	File read error
PATHERR	Invalid path name
NOMACH	String not found
PNOTFND	Path not found
WRDWRAP	Word too long to fit between fill margins - wrapped
INVMRG	Invalid fill margins
PNOTUNQ	Path not unique
PNOTWIN	Qualified name is not a window
FNOTOPN	Form is not open
NOHELP	No help available
NOHLPFRM	Error opening help form
INVRPT	Invalid repeat count
IMBBLK	Field must not contain blanks
BLKFLD	Field must be entered
INOTNUM	Field must be numeric
OUTRNG	Field value is out of range
ATNOTFND	Attribute does not exist
FINUSE	Can not close form - Form is in use
MFOPNERR	Message file can not be opened
INVKEY	Requested function can not be performed on this screen
INVIID	Invalid instance ID
NTMREJ	Network Transaction Manager (NTM) reject
MAPFAIL	Virtual terminal buffer format error
BUFOVFL	Virtual terminal buffer overflow
GTCURERR	GETCUR could not build fully qualified name
PARSERR	Level specified does not exist
PNOTITEM	Qualified name not an item
OLDFORM	Old form definition file format - Recompile
ENDARY	End of scrolling section reached
NOTSCROL	Field is not scrollable
PNOTBACK	Qualified name not a window or form
FILFRMMM	Form name does not match form definition file name
APKEY	Function key is to be processed by application
INTADDER	Internal error - Unable to add form to window

INDEB	In debug mode
OUTDEB	Out of debug mode
MSGRANGE	Message number specified does not exist
FLANERRS	Errors in Forms Language Compiler (FLAN) compilation
FDLOPERR	Error opening Forms Definition Language file
DUPFRM	FLAN compilation would produce duplicate open forms
EXITFDFE	Exit fdfe
FRMEXST	Form already exist
NFLDEXST	No field exists
NFLDMRKD	No field marked
LAYOUTER	Please correct errors marked before going to detailed mode
INTGETER	Internal error - Unable to get data from forms
INTRMVER	Internal error - Unable to remove form from window
ROLNCHG	Role not changed
WNDNSEL	Window not selected
UISWNNF	UIS window not found
SDPNDNG	Shut down pending
SDCNCLD	Shut down cancelled
INVRLFNC	Invalid role/function
NFPDSTRC	No FPD Structure found
NWNDIDFN	No unused window ID found
BADFPD	Bad logical device data structure - (Bad FPD)
WNDSEL	Window is selected
WMARCHSZ	Cannot change size of window which is an element in an array
WMARCHLC	Cannot change pos. of window which is an element in an array
SCPWNOPN	Script file could not be opened for writing
SCPRNOPN	Script file could not be opened for reading
SCPSNOPN	Script output save file could not be opened for writing
SCPBADFL	Script file is bad
SCPSVERR	Could not write to script output save file - close save file

SCPWRERR	Could not write to script file - terminating scripting
SCPRDERR	Could not read script file - terminating scripting session
OVRFLW	Adding an element would exceed bounds of form
BADDATA	Non-printable characters in data - replaced with "?"
WNDUNSEL	Window is unselected
APNOTFND	Application not found
CURFPDST	Cannot remove or close current logical device standalone
NINVTIMD	Not in Form Processor bypass mode (VTI mode)
INVTIMD	In Form Processor bypass mode (VTI mode)
INTPUTER	Internal Error - Could not put data into field
INTPATER	Internal Error - Could not charge attribute for a field
GTNAMERR	Could not find name
GTDQNERR	Could not find default qualified name
INTADLER	Internal Error - Could not add element to open-ended array
BADDEV	Unable to access specified device
BADAP	Unable to start specified application
BADAPS	Unable to create application data structures
FILWERR	File write error
BFOVRFLW	Data will not fit into buffer provided

The following errors are fatal errors:

LOGERR	Errors that are logged
NTMINT	Bad return from Network Transaction Manager (NTM) INITEX routine
NTMLOG	Bad return from NTM LOGON routine
NTMLO	Bad return from NTM LOGOFF routine
BSEND	Bad send from UI to AP
CROLER	Bad return from NTM CHGROL routine
UOPFRM	Bad return from OPEN FORM routine
NTMMSGNA	NTM message not accepted
NTMPRCNT	NTM terminate process message not accepted
FPABORT	Form Processor abort
USRABRT	User aborted
FLDTBIG	Field too big to be sent by the NTM and was truncated

☆U.S. Government Printing Office:1992—648-127/62199